activemq-cpp-3.4.1

Generated by Doxygen 1.8.0

Sat Apr 28 2012 14:22:58

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Data Structure Index

2.1 Class Hierarchy

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

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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. 649) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 741) to operate on.*

- class **CmsDestinationAccessor**

  *Extends the **CmsAccessor** (p. 635) to add support for resolving destination names.*

- class **CmsTemplate**

  ***CmsTemplate** (p. 649) 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. 649).*

- 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

Namespaces

- namespace **policies**

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. 145) class.*
- 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 **ActiveMQQueueBrowser**
- 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 **ActiveMQXAConnection**
- class **ActiveMQXAConnectionFactory**
- class **ActiveMQXASession**
- 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 **FifoMessageDispatchChannel**
- class **MessageDispatchChannel**
- class **PrefetchPolicy**

  Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.
- class **RedeliveryPolicy**

  *Interface for a **RedeliveryPolicy** (p. 1744) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*
- class **SimplePriorityMessageDispatchChannel**
- class **Synchronization**

  *Transacted Object **Synchronization** (p. 2056), used to sync the events of a Transaction with the items in the Transaction.*

5.5 activemq::core::policies Namespace Reference

Data Structures

- class **DefaultPrefetchPolicy**
- class **DefaultRedeliveryPolicy**

5.6 activemq::exceptions Namespace Reference

Data Structures

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

5.7 activemq::io Namespace Reference

Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**

  OutputStream filter that just logs the data being written.

5.8 activemq::library Namespace Reference

Data Structures

- class **ActiveMQCPP**

5.9 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. 675) that returns NULL for all calls.*
- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

5.10 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. 693).*
- class **CompositeTaskRunner**

  *A **Task** (p. 2073) 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 **Scheduler**

  Scheduler *(p. 1796) class for use in executing Runnable Tasks either periodically or one time only with optional delay.*
- class **SchedulerTimerTask**

  Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.
- class **Task**

  Represents a unit of work that requires one or more iterations to complete.
- class **TaskRunner**

5.11 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. 2167) interface, providing the base functionality that's common to most of the* ***TransportFactory*** *(p. 2167) instances.*
- class **CompositeTransport**

  A Composite ***Transport*** *(p. 2161) is a* ***Transport*** *(p. 2161) implementation that is composed of several Transports.*
- class **DefaultTransportListener**

  A Utility class that create empty implementations for the ***TransportListener*** *(p. 2176) interface so that a subclass only needs to override the one's its interested.*
- class **IOTransport**

  Implementation of the ***Transport*** *(p. 2161) 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. 2161) Factories that are available to the client at runtime.*

5.12 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.

5.13 activemq::transport::failover Namespace Reference

Data Structures

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

  *Creates an instance of a **FailoverTransport** (p. 1010).*
- class **FailoverTransportListener**

  *Utility class used by the **Transport** (p. 2161) to perform the work of responding to events from the active **Transport** (p. 2161).*
- class **URIPool**

5.14 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 {.

5.15 activemq::transport::logging Namespace Reference

Data Structures

- class **LoggingTransport**

  A transport filter that logs commands as they are sent/received.

5.16 activemq::transport::mock Namespace Reference

Data Structures

- class **InternalCommandListener**

  *Listens for Commands sent from the **MockTransport** (p. 1509).*
- class **MockTransport**

  *The **MockTransport** (p. 1509) defines a base level **Transport** (p. 2161) class that is intended to be used in place of an a regular protocol **Transport** (p. 2161) suck 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.

5.17 activemq::transport::tcp Namespace Reference

Data Structures

- class **SslTransport**

  Transport *(p. 2161) for connecting to a Broker using an SSL Socket.*
- class **SslTransportFactory**
- class **TcpTransport**

  Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an ***IOTransport*** *(p. 1200).*
- class **TcpTransportFactory**

  Factory Responsible for creating the ***TcpTransport*** *(p. 2086).*

5.18 activemq::util Namespace Reference

Data Structures

- class **ActiveMQProperties**

  Implementation of the CMSProperties interface that delegates to a ***decaf::util::Properties*** *(p. 1705) object.*
- class **CMSExceptionSupport**
- class **CompositeData**

  Represents a Composite URI.
- class **IdGenerator**
- 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 **MarshallingSupport**
- 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. 1662) from one type to another.*
- class **PrimitiveValueNode**

  Class that wraps around a single value of one of the many types.
- class **Service**

  Base interface for all classes that run as a ***Service*** *(p. 1826) inside the application.*
- class **ServiceListener**

  Listener interface for observers of ***Service*** *(p. 1826) related events.*
- class **ServiceStopper**
- class **ServiceSupport**

  Provides a base class for ***Service*** *(p. 1826) implementations.*
- class **URISupport**
- class **Usage**

5.19 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. 2239) is the interface that all **WireFormatFactory** (p. 2239) classes must extend.*
- class **WireFormatNegotiator**

  *Defines a **WireFormatNegotiator** (p. 2251) which allows a **WireFormat** (p. 2236) to.*
- class **WireFormatRegistry**

  *Registry of all **WireFormat** (p. 2236) Factories that are available to the client at runtime.*

5.20 activemq::wireformat::openwire Namespace Reference

Namespaces

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

Data Structures

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

  Used to allow a MockTransport to generate response commands to OpenWire Commands.

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

Namespaces

- namespace **generated**

Data Structures

- class **BaseDataStreamMarshaller**

  Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire 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.

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

Data Structures

- class **ActiveMQBlobMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 126).*
- class **ActiveMQBytesMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 142).*
- class **ActiveMQDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 200).*
- class **ActiveMQMapMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 220).*
- class **ActiveMQMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 225).*
- class **ActiveMQObjectMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 235).*
- class **ActiveMQQueueMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 255).*
- class **ActiveMQStreamMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 290).*
- class **ActiveMQTempDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 296).*
- class **ActiveMQTempQueueMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 303).*
- class **ActiveMQTempTopicMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 310).*
- class **ActiveMQTextMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 318).*
- class **ActiveMQTopicMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 324).*
- class **BaseCommandMarshaller**

  *Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 386).*
- class **BrokerIdMarshaller**

  *Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 440).*
- class **BrokerInfoMarshaller**

  *Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 448).*
- class **ConnectionControlMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 732).*
- class **ConnectionErrorMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 738).*
- class **ConnectionIdMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 748).*
- class **ConnectionInfoMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 756).*
- class **ConsumerControlMarshaller**

  *Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 773).*
- class **ConsumerIdMarshaller**

  *Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 779).*
- class **ConsumerInfoMarshaller**

  *Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 789).*
- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 795).*

- class **DataArrayResponseMarshaller**

  *Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 831).*
- class **DataResponseMarshaller**

  *Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 865).*
- class **DestinationInfoMarshaller**

  *Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 943).*
- class **DiscoveryEventMarshaller**

  *Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 952).*
- class **ExceptionResponseMarshaller**

  *Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 999).*
- class **FlushCommandMarshaller**

  *Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1070).*
- class **IntegerResponseMarshaller**

  *Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1177).*
- class **JournalQueueAckMarshaller**

  *Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1213).*
- class **JournalTopicAckMarshaller**

  *Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1219).*
- class **JournalTraceMarshaller**

  *Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1224).*
- class **JournalTransactionMarshaller**

  *Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1230).*
- class **KeepAliveInfoMarshaller**

  *Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1236).*
- class **LastPartialCommandMarshaller**

  *Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1247).*
- class **LocalTransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1301).*
- 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. 1452).*
- class **MessageDispatchMarshaller**

  *Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1466).*
- class **MessageDispatchNotificationMarshaller**

  *Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1472).*
- class **MessageIdMarshaller**

  *Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1482).*
- class **MessageMarshaller**

  *Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1486).*
- class **MessagePullMarshaller**

  *Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1506).*
- class **NetworkBridgeFilterMarshaller**

  *Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1530).*
- class **PartialCommandMarshaller**

  *Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 1611).*
- class **ProducerAckMarshaller**

  *Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 1686).*
- class **ProducerIdMarshaller**

  *Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 1694).*

- class **ProducerInfoMarshaller**

  *Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 1701).*
- class **RemoveInfoMarshaller**

  *Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 1761).*
- class **RemoveSubscriptionInfoMarshaller**

  *Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 1767).*
- class **ReplayCommandMarshaller**

  *Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 1773).*
- class **ResponseMarshaller**

  *Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 1788).*
- class **SessionIdMarshaller**

  *Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 1846).*
- class **SessionInfoMarshaller**

  *Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 1851).*
- class **ShutdownInfoMarshaller**

  *Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 1886).*
- class **SubscriptionInfoMarshaller**

  *Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2042).*
- class **TransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2146).*
- class **TransactionInfoMarshaller**

  *Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2152).*
- class **WireFormatInfoMarshaller**

  *Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2248).*
- class **XATransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2281).*

5.23 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. 1089) class maps hexidecimal 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.

5.24 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** (p. 2005)'s.*
- class **StompWireFormat**
- class **StompWireFormatFactory**

  Factory used to create the Stomp Wire Format instance.

5.25 cms Namespace Reference

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

Data Structures

- class **BytesMessage**

  *A **BytesMessage** (p. 557) 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. 725) objects returned implement the CMS **Connection** (p. 725) interface and hide the CMS Provider specific implementation details behind that interface.*
- class **ConnectionMetaData**

  *A **ConnectionMetaData** (p. 759) object provides information describing the **Connection** (p. 725) 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. 936) 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. 996) that is registered with the **Connection** (p. 725).*
- 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. 1379) 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. 1455) to received messages from a destination.*
- class **MessageEnumeration**

  Defines an object that enumerates a collection of Messages.
- class **MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2020) or **BytesMessage** (p. 557) 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. 1485) 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. 1491) object to send messages to a **Destination** (p. 936).*

- 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. 1722) without removing them.*

- class **Session**

  *A **Session** (p. 1830) 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. 2020).*

- class **TemporaryQueue**

  *Defines a Temporary **Queue** (p. 1722) based **Destination** (p. 936).*

- class **TemporaryTopic**

  *Defines a Temporary **Topic** (p. 2141) based **Destination** (p. 936).*

- class **TextMessage**

  Interface for a text message.

- class **Topic**

  An interface encapsulating a provider-specific topic name.

- class **TransactionInProgressException**

  This exception is thrown when an operation is invalid because a transaction is in progress.

- class **TransactionRolledBackException**

  *This exception must be thrown when a call to **Session.commit** (p. 1834) results in a rollback of the current transaction.*

- 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.

- class **XAConnection**

  *The **XAConnection** (p. 2262) interface defines an extended **Connection** (p. 725) type that is used to create **XASession** (p. 2275) objects.*

- class **XAConnectionFactory**

  *The **XAConnectionFactory** (p. 2263) interface is specialized interface that defines an **ConnectionFactory** (p. 741) that creates **Connection** (p. 725) instance that will participate in XA Transactions.*

- class **XAException**

  *The **XAException** (p. 2265) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.*

- class **XAResource**

  *The **XAResource** (p. 2269) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).*
- class **XASession**

  *The **XASession** (p. 2275) interface extends the capability of **Session** (p. 1830) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).*
- class **Xid**

  An interface which provides a mapping for the X/Open XID transaction identifier structure.

5.25.1 Detailed Description

5.26 decaf Namespace Reference

Namespaces

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

5.26.1 Detailed Description

5.27 decaf::internal Namespace Reference

Namespaces

- namespace **io**

- namespace **net**
- namespace **nio**
- namespace **security**
- 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.

5.28 decaf::internal::io Namespace Reference

Data Structures

- class **StandardErrorOutputStream**

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

5.29 decaf::internal::net Namespace Reference

Namespaces

- namespace **ssl**
- namespace **tcp**

Data Structures

- class **DefaultServerSocketFactory**

  Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.
- class **DefaultSocketFactory**

  SocketFactory implementation that is used to create Sockets.
- class **Network**

  Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.
- class **SocketFileDescriptor**

  File Descriptor type used internally by Decaf Socket objects.
- 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.

5.30 decaf::internal::net::ssl Namespace Reference

Namespaces

- namespace **openssl**

Data Structures

- class **DefaultSSLContext**

  Default SSLContext manager for the Decaf library.
- class **DefaultSSLServerSocketFactory**

  Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.
- class **DefaultSSLSocketFactory**

  Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

5.31 decaf::internal::net::ssl::openssl Namespace Reference

Data Structures

- class **OpenSSLContextSpi**

  Provides an SSLContext that wraps the OpenSSL API.
- class **OpenSSLParameters**

  Container class for parameters that are Common to OpenSSL socket classes.
- class **OpenSSLServerSocket**

  SSLServerSocket based on OpenSSL library code.
- class **OpenSSLServerSocketFactory**

  SSLServerSocketFactory that creates Server Sockets that use OpenSSL.
- class **OpenSSLSocket**

  Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.
- class **OpenSSLSocketException**

  Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.
- class **OpenSSLSocketFactory**

  Client Socket Factory that creates SSL based client sockets using the OpenSSL library.
- class **OpenSSLSocketInputStream**

  An output stream for reading data from an OpenSSL Socket instance.
- class **OpenSSLSocketOutputStream**

  *OutputStream implementation used to write data to an **OpenSSLSocket** (p. 1560) instance.*

5.32 decaf::internal::net::tcp Namespace Reference

Data Structures

- class **TcpSocket**

  Platform-independent implementation of the socket interface.
- class **TcpSocketInputStream**

  Input stream for performing reads on a socket.
- class **TcpSocketOutputStream**

  Output stream for performing write operations on a socket.

5.33 decaf::internal::nio Namespace Reference

Data Structures

- class **BufferFactory**

  *Factory class used by static methods in the **decaf::nio** (p. 74) package to create the various default version of the NIO interfaces.*

- class **ByteArrayBuffer**

  This class defines six categories of operations upon byte buffers:

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

5.34 decaf::internal::security Namespace Reference

Data Structures

- class **SecureRandomImpl**

  Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

5.35 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 **GenericResource**

  *A Generic **Resource** (p. 1777) wraps some type and will delete it when the **Resource** (p. 1777) itself is deleted.*

- class **HexStringParser**
- class **Resource**

  Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

- class **ResourceLifecycleManager**
- class **TimerTaskHeap**

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

5.36 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**

5.37 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**

  *A **ByteArrayInputStream** (p. 527) contains an internal buffer that contains bytes that may be read from the stream.*
- class **ByteArrayOutputStream**
- class **Closeable**

  Interface for a class that implements the close method.
- class **DataInput**

  *The **DataInput** (p. 842) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*
- 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 **DataOutput**

  *The **DataOutput** (p. 856) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*
- 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 **FileDescriptor**

  This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.
- class **FilterInputStream**

  *A **FilterInputStream** (p. 1032) 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 **Flushable**

  *A **Flushable** (p. 1067) is a destination of data that can be flushed.*
- class **InputStream**

  A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.
- class **InputStreamReader**

  *An **InputStreamReader** (p. 1142) is a bridge from byte streams to character streams.*
- class **InterruptedIOException**
- class **IOException**
- class **OutputStream**

  Base interface for any class that wants to represent an output stream of bytes.
- class **OutputStreamWriter**

  A class for turning a character stream into a byte stream.
- class **PushbackInputStream**

  *A **PushbackInputStream** (p. 1716) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*
- 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**

5.38 decaf::lang Namespace Reference

Namespaces

- namespace **exceptions**

Data Structures

- class **Appendable**

  An object to which char sequences and values can be appended.
- class **ArrayPointer**

  *Decaf's implementation of a Smart **Pointer** (p. 1614) that is a template on a Type and is **Thread** (p. 2094) Safe if the default Reference Counter is used.*
- class **ArrayPointerComparator**

  *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 value of the actual pointer to the array being contained in this **ArrayPointer** (p. 358).*
- class **Boolean**
- class **Byte**
- class **Character**
- class **CharSequence**

  *A **CharSequence** (p. 623) 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. 1207) type for generic collections API calls.*

- class **Long**
- class **Math**

  *The class **Math** (p. 1396) 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. 1543) is the superclass of classes **Byte** (p. 476), **Double** (p. 956), **Float** (p. 1040), **Integer** (p. 1161), **Long** (p. 1338), and **Short** (p. 1858).*
- struct **STATIC_CAST_TOKEN**
- struct **DYNAMIC_CAST_TOKEN**
- class **Pointer**

  *Decaf's implementation of a Smart **Pointer** (p. 1614) that is a template on a Type and is **Thread** (p. 2094) 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. 1614) instance.*
- class **Readable**

  *A **Readable** (p. 1732) is a source of characters.*
- class **Runnable**

  Interface for a runnable object - defines a task that can be run by a thread.
- class **Runtime**
- class **Short**
- class **String**

  *The **String** (p. 2031) class represents an immutable sequence of chars.*
- class **System**

  *The **System** (p. 2065) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*
- class **Thread**

  *A **Thread** (p. 2094) is a concurrent unit of execution.*
- class **ThreadGroup**
- class **Throwable**

  This class represents an error that has occurred.

Functions

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

5.38.1 Function Documentation

5.38.1.1 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().

5.38.1.2 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().

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

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

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

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

5.38.1.5 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().

5.38.1.6 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().

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

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

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

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

5.39 decaf::lang::exceptions Namespace Reference

Data Structures

- class **ClassCastException**
- class **IllegalArgumentException**
- class **IllegalMonitorStateException**
- class **IllegalStateException**
- class **IllegalThreadStateException**

- class **IndexOutOfBoundsException**
- class **InterruptedException**
- class **InvalidStateException**
- class **NullPointerException**
- class **NumberFormatException**
- class **RuntimeException**
- class **UnsupportedOperationException**

5.40 decaf::net Namespace Reference

Namespaces

- namespace **ssl**

Data Structures

- class **BindException**
- class **ConnectException**
- class **DatagramPacket**

  Class that represents a single datagram packet.
- class **HttpRetryException**
- class **Inet4Address**
- class **Inet6Address**
- class **InetAddress**

  Represents an IP address.
- class **InetSocketAddress**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**

  This class implements server sockets.
- class **ServerSocketFactory**

  Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.
- class **Socket**
- class **SocketAddress**

  *Base class for protocol specific **Socket*** (p. *1900*) *addresses.*
- class **SocketError**

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

  Exception for errors when manipulating sockets.
- class **SocketFactory**

  *The **SocketFactory*** (p. *1916*) *is used to create **Socket*** (p. *1900*) *objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*
- class **SocketImpl**

  *Acts as a base class for all physical **Socket*** (p. *1900*) *implementations.*
- class **SocketImplFactory**

  Factory class interface for a Factory that creates ScoketImpl objects.
- class **SocketOptions**
- class **SocketTimeoutException**

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

  *This class represents an instance of a **URI** (p. 2191) as defined by RFC 2396.*
- class **URISyntaxException**
- class **URL**

  *Class **URL** (p. 2223) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*
- class **URLDecoder**
- class **URLEncoder**

5.41 decaf::net::ssl Namespace Reference

Data Structures

- class **SSLContext**

  *Represents on implementation of the Secure **Socket** (p. 1900) Layer for streaming based sockets.*
- class **SSLContextSpi**

  *Defines the interface that should be provided by an **SSLContext** (p. 1934) provider.*
- class **SSLParameters**
- class **SSLServerSocket**

  Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.
- class **SSLServerSocketFactory**

  Factory class interface that provides methods to create SSL Server Sockets.
- class **SSLSocket**
- class **SSLSocketFactory**

  *Factory class interface for a **SocketFactory** (p. 1916) that can create **SSLSocket** (p. 1947) objects.*

5.42 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:

5.43 decaf::security Namespace Reference

Namespaces

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

Data Structures

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

  *The **Key** (p. 1239) interface is the top-level interface for all keys.*
- class **KeyException**
- class **KeyManagementException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

  Base interface for a principal, which can represent an individual or organization.
- class **PublicKey**

  A public key.
- class **SecureRandom**
- class **SecureRandomSpi**

  Interface class used by Security Service Providers to implement a source of secure random bytes.
- class **SignatureException**

5.44 decaf::security::auth Namespace Reference

Namespaces

- namespace **x500**

5.45 decaf::security::auth::x500 Namespace Reference

Data Structures

- class **X500Principal**

5.46 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.

5.47 decaf::util Namespace Reference

Namespaces

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

Data Structures

- class **AbstractCollection**

  *This class provides a skeletal implementation of the **Collection** (p. 660) interface, to minimize the effort required to implement this interface.*
- class **AbstractList**

  *This class provides a skeletal implementation of the **List** (p. 1286) 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. 1371) interface, to minimize the effort required to implement this interface.*
- class **AbstractQueue**

  *This class provides skeletal implementations of some **Queue** (p. 1723) operations.*
- class **AbstractSequentialList**

  *This class provides a skeletal implementation of the **List** (p. 1286) 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. 1857) interface to minimize the effort required to implement this interface.*
- class **ArrayList**
- class **Arrays**
- 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 **ConcurrentModificationException**
- class **Date**

  Wrapper class around a time value in milliseconds.
- class **Deque**

  *Defines a 'Double ended **Queue** (p. 1723)' interface that allows for insertion and removal of elements from both ends.*
- class **Iterator**

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

  *A complete implementation of the **List** (p. 1286) interface using a doubly linked list data structure.*
- 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. 1371) 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 **NoSuchElementException**
- 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. *1728) 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. *1286) 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. *1371) 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. *1723) class accepts messages with an psuh(m) command where m is the message to be queued.*

- class **StlSet**

  Set (p. *1857) 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 that allows for parsing of string based on Tokens.

- 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. *2122).*

- class **UUID**

  *A class that represents an immutable universally unique identifier (**UUID*** (p. *2230)).*

5.48 decaf::util::comparators Namespace Reference

Data Structures

- class **Less**

  *Simple **Less*** (p. *1250) **Comparator*** (p. *689) that compares to elements to determine if the first is less than the second.*

5.49 decaf::util::concurrent Namespace Reference

Namespaces

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

Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **AbstractExecutorService**

  *Provides a default implementation for the methods of the **ExecutorService** (p. 1008) interface.*
- class **BlockingQueue**

  *A **decaf::util::Queue** (p. 1723) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*
- class **BrokenBarrierException**
- class **Callable**

  A task that returns a result and may throw an exception.
- class **CancellationException**
- class **ConcurrentMap**

  *Interface for a **Map** (p. 1371) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1371) interface.*
- class **ConcurrentStlMap**

  ***Map** (p. 1371) 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 **CopyOnWriteArrayList**
- class **CopyOnWriteArraySet**

  *Since the **CopyOnWriteArraySet** (p. 813) and the **CopyOnWriteArrayList** (p. 798) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 798) for all its underlying operations.*
- 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. 1792) tasks.*
- class **Executors**

  *Implements a set of utilities for use with **Executors** (p. 1005), **ExecutorService** (p. 1008), **ThreadFactory** (p. 2102), and **Callable** (p. 578) types, as well as providing factory methods for instance of these types configured for the most common use cases.*
- class **ExecutorService**

  *An **Executor** (p. 1004) that provides methods to manage termination and methods that can produce a **Future** (p. 1075) for tracking progress of one or more asynchronous tasks.*
- class **Future**

  *A **Future** (p. 1075) represents the result of an asynchronous computation.*
- class **LinkedBlockingQueue**

  *A **BlockingQueue** (p. 417) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.*
- class **Lock**

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

  ***Mutex** (p. 1519) 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 **RejectedExecutionException**
- class **RejectedExecutionHandler**

  *A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. 2104).*
- 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 **blocking queue** (p. 417) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*

- class **ThreadFactory**

  *public interface **ThreadFactory** (p. 2102)*

- class **ThreadPoolExecutor**

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

- class **TimeoutException**

- class **TimeUnit**

  *A **TimeUnit** (p. 2134) 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.*

5.50 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 **AtomicRefCounter**

- class **AtomicReference**

  An Pointer reference that may be updated atomically.

5.51 decaf::util::concurrent::locks Namespace Reference

Data Structures

- class **AbstractOwnableSynchronizer**

  Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

- class **Condition**

  ***Condition** (p. 715) factors out the **Mutex** (p. 1519) 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. 1305) implementations.*

- class **Lock**

  ***Lock** (p. 1305) 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. 1741) maintains a pair of associated locks, one for read-only operations and one for writing.*

- class **ReentrantLock**

  *A reentrant mutual exclusion **Lock** (p. 1305) with extended capabilities.*

5.52 decaf::util::logging Namespace Reference

Data Structures

- class **ConsoleHandler**

  *This **Handler** (p. 1085) publishes log records to System.err.*

- class **ErrorManager**

  ***ErrorManager** (p. 988) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1085) during Logging.*

- class **Filter**

  *A **Filter** (p. 1031) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

- class **Formatter**

  *A **Formatter** (p. 1074) provides support for formatting LogRecords.*

- class **Handler**

  *A **Handler** (p. 1085) object takes log messages from a **Logger** (p. 1312) and exports them.*

- class **Level**

  *The **Level** (p. 1253) class defines a set of standard logging levels that can be used to control logging output.*

- class **Logger**

  *A **Logger** (p. 1312) object is used to log messages for a specific system or application component.*

- class **LoggerHierarchy**

- class **LogManager**

  *There is a single global **LogManager** (p. 1327) object that is used to maintain a set of shared state about Loggers and log services.*

- class **LogRecord**

  ***LogRecord** (p. 1333) objects are used to pass logging requests between the logging framework and individual log Handlers.*

- 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. 1705).*

- class **SimpleFormatter**

  *Print a brief summary of the **LogRecord** (p. 1333) in a human readable format.*

- class **SimpleLogger**

- class **StreamHandler**

  *Stream based logging **Handler** (p. 1085).*

- class **XMLFormatter**

  *Format a **LogRecord** (p. 1333) into a standard XML format.*

Enumerations

- enum **Levels** {
  Off, **Null**, **Markblock**, **Debug**,
  Info, **Warn**, **Error**, **Fatal**,
  Throwing }

  Defines an enumeration for logging levels.

5.52.1 Enumeration Type Documentation

5.52.1.1 enum decaf::util::logging::Levels

Defines an enumeration for logging levels.

Enumerator:

Off

Null

Markblock

Debug

Info

Warn

Error

Fatal

Throwing

5.53 decaf::util::zip Namespace Reference

Data Structures

- class **Adler32**

  Clas that can be used to compute an Adler-32 ***Checksum*** *(*p. *628) for a data stream.*

- class **CheckedInputStream**

  An implementation of a FilterInputStream that will maintain a ***Checksum*** *(*p. *628) of the bytes read, the* ***Checksum*** *(*p. *628) can then be used to verify the integrity of the input stream.*

- class **CheckedOutputStream**

  An implementation of a FilterOutputStream that will maintain a ***Checksum*** *(*p. *628) of the bytes written, the* ***Checksum*** *(*p. *628) can then be used to verify the integrity of the output stream.*

- class **Checksum**

  An interface used to represent ***Checksum*** *(*p. *628) values in the Zip package.*

- class **CRC32**

  Class that can be used to compute a CRC-32 checksum for a data stream.

- class **DataFormatException**

- class **Deflater**

  This class compresses data using the DEFLATE algorithm (see `specification`*).*

- class **DeflaterOutputStream**

  Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

- class **Inflater**

  This class uncompresses data that was compressed using the DEFLATE algorithm (see `specification`*).*

- class **InflaterInputStream**

  A FilterInputStream that decompresses data read from the wrapped InputStream instance.

- class **ZipException**

5.54 std Namespace Reference

Data Structures

- struct **less**< **decaf::lang::ArrayPointer**< **T** > >

  An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.
- 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::concurrent::ThreadPoolExecutor::AbortPolicy Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always throws a **RejectedExecutionException** (p. 1755).

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

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy:

Public Member Functions

- **AbortPolicy** ()
- virtual ∼**AbortPolicy** ()
- virtual void **rejectedExecution** (**decaf::lang::Runnable** ∗task, **ThreadPoolExecutor** ∗executer **DECAF_UNUSED**)

6.1.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always throws a **RejectedExecutionException** (p. 1755).

Since

1.0

6.1.2 Constructor & Destructor Documentation

6.1.2.1 decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy::AbortPolicy ( ) [inline]

6.1.2.2 virtual decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy::∼AbortPolicy ( ) [inline, virtual]

6.1.3 Member Function Documentation

6.1.3.1 virtual void decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy::rejectedExecution (decaf::lang::Runnable * *task*, ThreadPoolExecutor *executer *DECAF_UNUSED*) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

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

This class provides a skeletal implementation of the **Collection** (p. 660) 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

- **AbstractCollection** ()
- 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 void **clear** ()

  Removes all of the elements from this collection (optional operation).
- virtual bool **contains** (**const** E &value) **const**

  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 (value == NULL ? e == NULL : value == e).

  Parameters

value	*The value to check for presence in the collection.*

  Returns

  true if there is at least one of the elements in the collection

  Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

- virtual bool **containsAll** (**const Collection**< E > &collection) **const**

  Returns true if this collection contains all of the elements in the specified collection.

  Parameters

collection	*The **Collection** (p. 660) to compare to this one.*

  Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

- virtual bool **equals** (**const Collection**< E > &collection) **const**

  *Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.*
- virtual void **copy** (**const Collection**< E > &collection)

  *Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).*

- virtual bool **isEmpty** () **const**

  Returns true if this collection contains no elements.

- virtual bool **add** (**const** E &value **DECAF_UNUSED**)

- virtual bool **addAll** (**const Collection**< E > &collection)

  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	*The **Collection** (p. 660) whose elements are added to this one.*

  Returns

  true if this collection changed as a result of the call

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
IllegalArgumentException	*if some property of an element prevents it from being added to this collection*
IllegalStateException	*if an element cannot be added at this time due to insertion restrictions.*

- virtual bool **remove** (**const** E &value)

  Removes a single instance of the specified element from the collection.
  More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this **Collection** (p. 660).*

  Returns

  true if the collection was changed, false otherwise.

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

- virtual bool **removeAll** (**const Collection**< E > &collection)

  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. 660) whose elements are to be removed from this one.*

  Returns

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

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

- virtual bool **retainAll** (**const Collection**< E > &collection)

  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. 660) whose elements are to be retained.*

  Returns

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

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

- virtual std::vector< E > **toArray** () **const**

  *Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).*

- virtual void **lock** ()

  Locks the object.

- virtual bool **tryLock** ()

  Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

  Unlocks the object.

- virtual void **wait** ()

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

- virtual void **wait** (long long millisecs)

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

- virtual void **wait** (long long millisecs, int nanos)

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

- virtual void **notify** ()

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

- virtual void **notifyAll** ()

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

Protected Attributes

- **util::concurrent::Mutex mutex**

6.2.1 Detailed Description

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

This class provides a skeletal implementation of the **Collection** (p. 660) 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. 660) constructor, as per the recommendation in the **Collection** (p. 660) 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

6.2.2 Constructor & Destructor Documentation

6.2.2.1 template<typename E> decaf::util::AbstractCollection< E >::AbstractCollection ( ) [inline]

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

6.2.3 Member Function Documentation

6.2.3.1 template<typename E> virtual bool decaf::util::AbstractCollection< E >::add (const E &value *DECAF_UNUSED*) [inline, virtual]

This implementation always throws an UnsupportedOperationException.

Referenced by decaf::util::AbstractCollection< cms::Connection * >::addAll(), decaf::util::AbstractCollection< cms::Connection * >::copy(), and decaf::util::AbstractCollection< cms::Connection * >::operator=().

6.2.3.2 template<typename E> virtual bool decaf::util::AbstractCollection< E >::addAll (const Collection< E > & *collection*) [inline, 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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

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).

Implements **decaf::util::Collection< E >** (p. 663).

Reimplemented in **decaf::util::StlList< E >** (p. 1971), **decaf::util::ArrayList< E >** (p. 349), **decaf::util::LinkedList< E >** (p. 1272), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1272), **decaf::util::LinkedList< CompositeTask * >** (p. 1272), **decaf::util::LinkedList< URI >** (p. 1272), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1272), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1272), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1272), **decaf::util::LinkedList< Pointer< Command > >** (p. 1272), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1272), **decaf::util::LinkedList< cms::Destination * >** (p. 1272), **decaf::util::LinkedList< cms::Session * >** (p. 1272), **decaf::util::LinkedList< cms::Connection * >** (p. 1272), **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 817),

and **decaf::util::AbstractQueue**< **E** > (p. 110).

6.2.3.3 template<typename E> virtual void decaf::util::AbstractCollection< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

Implements **decaf::util::Collection**< **E** > (p. 663).

Reimplemented in **decaf::util::AbstractList**< **E** > (p. 100), **decaf::util::AbstractList**< **Pointer**< **Transport** > > (p. 100), **decaf::util::AbstractList**< **cms::MessageConsumer** * > (p. 100), **decaf::util::AbstractList**< **CompositeTask** * > (p. 100), **decaf::util::AbstractList**< **URI** > (p. 100), **decaf::util::AbstractList**< **Pointer**< **MessageDispatch** > > (p. 100), **decaf::util::AbstractList**< **Pointer**< **DestinationInfo** > > (p. 100), **decaf::util::AbstractList**< **PrimitiveValueNode** > (p. 100), **decaf::util::AbstractList**< **Pointer**< **Command** > > (p. 100), **decaf::util::AbstractList**< **Pointer**< **BackupTransport** > > (p. 100), **decaf::util::AbstractList**< **cms::MessageProducer** * > (p. 100), **decaf::util::AbstractList**< **cms::Destination** * > (p. 100), **decaf::util::AbstractList**< **cms::Session** * > (p. 100), **decaf::util::AbstractList**< **cms::Connection** * > (p. 100), **decaf::util::StlList**< **E** > (p. 1972), **decaf::util::PriorityQueue**< **E** > (p. 1679), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2059), **decaf::util::concurrent::LinkedBlockingQueue**< **E** > (p. 1260), **decaf::util::LinkedList**< **E** > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1274), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1274), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1274), **decaf::util::LinkedList**< **URI** > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1274), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1274), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1274), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1274), **decaf::util::LinkedList**< **cms::Session** * > (p. 1274), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1274), **decaf::util::StlSet**< **E** > (p. 1998), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 1998), **decaf::util::StlSet**< **Resource** * > (p. 1998), **decaf::util::ArrayList**< **E** > (p. 350), **decaf::util::AbstractQueue**< **E** > (p. 110), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 818).

Referenced by decaf::util::AbstractCollection< cms::Connection * >::copy(), and decaf::util::AbstractCollection< cms::Connection * >::operator=().

6.2.3.4 template<typename E> virtual bool decaf::util::AbstractCollection< E >::contains (const E & *value*) const [inline, 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 (value == NULL ? e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Implements **decaf::util::Collection< E >** (p. 664).

Reimplemented in **decaf::util::StlList< E >** (p. 1972), **decaf::util::ArrayList< E >** (p. 350), **decaf::util::LinkedList< E >** (p. 1275), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1275), **decaf::util::LinkedList< CompositeTask * >** (p. 1275), **decaf::util::LinkedList< URI >** (p. 1275), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1275), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1275), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1275), **decaf::util::LinkedList< Pointer< Command > >** (p. 1275), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1275), **decaf::util::LinkedList< cms::Destination * >** (p. 1275), **decaf::util::LinkedList< cms::Session * >** (p. 1275), **decaf::util::LinkedList< cms::Connection * >** (p. 1275), **decaf::util::StlSet< E >** (p. 1998), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 1998), **decaf::util::StlSet< Resource * >** (p. 1998), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 818).

Referenced by decaf::util::AbstractCollection< cms::Connection * >::containsAll().

6.2.3.5 template<typename E> virtual bool decaf::util::AbstractCollection< E >::containsAll (const Collection< E > & *collection*) const [inline, virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	The **Collection** (p. 660) to compare to this one.

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

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.

Implements **decaf::util::Collection< E >** (p. 665).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 2059), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 818).

Referenced by decaf::util::AbstractCollection< cms::Connection * >::equals().

6.2.3.6 template<typename E> virtual void decaf::util::AbstractCollection< E >::copy (const Collection< E > & *collection*) [inline, virtual]

Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 660)'s clear method.

Parameters

collection	The collection to mirror.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
IllegalStateException	if the elements cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection**< **E** > (p. 666).

Reimplemented in **decaf::util::StlList**< **E** > (p. 1973), **decaf::util::LinkedList**< **E** > (p. 1275), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1275), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1275), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1275), **decaf::util::LinkedList**< **URI** > (p. 1275), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1275), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1275), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1275), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1275), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1275), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1275), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1275), **decaf::util::LinkedList**< **cms::Session** * > (p. 1275), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1275), **decaf::util::StlSet**< **E** > (p. 1999), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 1999), **decaf::util::StlSet**< **Resource** * > (p. 1999), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 819).

6.2.3.7 template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals (const Collection< E > & *collection*) const [inline, virtual]

Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.

Parameters

collection	- The **Collection** (p. 660) to be compared to this one.

Returns

true if this **Collection** (p. 660) is equal to the one given.

Implements **decaf::util::Collection**< **E** > (p. 666).

Reimplemented in **decaf::util::StlList**< **E** > (p. 1973), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2061), **decaf::util::StlSet**< **E** > (p. 1999), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 1999), **decaf::util::StlSet**< **Resource** * > (p. 1999), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 819).

6.2.3.8 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. 669) == 0.

Returns

true if the size method return 0.

Implements **decaf::util::Collection**< **E** > (p. 667).

Reimplemented in **decaf::util::StlList**< **E** > (p. 1974), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2061), **decaf::util::LinkedList**< **E** > (p. 1278), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1278),

decaf::util::LinkedList< cms::MessageConsumer * > (p. 1278), **decaf::util::LinkedList< CompositeTask * >** (p. 1278), **decaf::util::LinkedList< URI >** (p. 1278), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1278), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1278), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1278), **decaf::util::LinkedList< Pointer< Command > >** (p. 1278), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1278), **decaf::util::LinkedList< cms::Destination * >** (p. 1278), **decaf::util::LinkedList< cms::Session * >** (p. 1278), **decaf::util::LinkedList< cms::Connection * >** (p. 1278), **decaf::util::StlSet< E >** (p. 1999), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 1999), **decaf::util::StlSet< Resource * >** (p. 1999), **decaf::util::ArrayList< E >** (p. 352), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 820).

Referenced by decaf::util::AbstractQueue< E >::clear(), decaf::util::PriorityQueue< E >::peek(), decaf::util::PriorityQueue< E >::poll(), and decaf::util::PriorityQueue< E >::remove().

6.2.3.9 template<typename E> virtual void decaf::util::AbstractCollection< E >::lock ( ) [inline, virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::clear(), decaf::util::concurrent::LinkedBlockingQueue< E >::remove(), and decaf::util::concurrent::LinkedBlockingQueue< E >::toArray().

6.2.3.10 template<typename E> virtual void decaf::util::AbstractCollection< E >::notify ( ) [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. 2047).

6.2.3.11 template<typename E> virtual void decaf::util::AbstractCollection< E >::notifyAll ( ) [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. 2048).

6.2.3.12 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

6.2.3.13 template<typename E> virtual bool decaf::util::AbstractCollection< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

Implements **decaf::util::Collection< E >** (p. 667).

Reimplemented in **decaf::util::StlList< E >** (p. 1976), **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1265), **decaf::util::PriorityQueue< E >** (p. 1682), **decaf::util::ArrayList< E >** (p. 353), **decaf::util::StlSet< E >** (p. 2000), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2000), **decaf::util::StlSet< Resource * >** (p. 2000), **decaf::util::LinkedList< E >** (p. 1283), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1283), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1283), **decaf::util::LinkedList< CompositeTask * >** (p. 1283), **decaf::util::LinkedList< URI >** (p. 1283), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1283), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1283), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1283), **decaf::util::LinkedList< Pointer< Command > >** (p. 1283), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1283), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1283), **decaf::util::LinkedList< cms::Destination * >** (p. 1283), **decaf::util::LinkedList< cms::Session * >** (p. 1283), **decaf::util::LinkedList< cms::Connection * >** (p. 1283), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 820).

6.2.3.14 template<typename E> virtual bool decaf::util::AbstractCollection< E >::removeAll (const Collection< E > & *collection*) [inline, 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. 660) whose elements are to be removed from this one.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

Implements **decaf::util::Collection**< **E** > (p. 668).

Reimplemented in **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 821), **decaf::util::AbstractSet**< **E** > (p. 119), **decaf::util::AbstractSet**< **Pointer**< **Synchronization** > > (p. 119), and **decaf::util::AbstractSet**< **Resource** * > (p. 119).

6.2.3.15 template<typename E> virtual bool decaf::util::AbstractCollection< E >::retainAll (const Collection< E > & *collection*) [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.

Parameters

collection	The **Collection** (p. 660) whose elements are to be retained.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

Implements **decaf::util::Collection< E >** (p. 669).

Reimplemented in **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 821).

6.2.3.16 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. 660).

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. 660)

Implements **decaf::util::Collection< E >** (p. 670).

Reimplemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1266), **decaf::util::ArrayList< E >** (p. 355), **decaf::util::LinkedList< E >** (p. 1286), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1286), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1286), **decaf::util::LinkedList< CompositeTask * >** (p. 1286), **decaf::util::LinkedList< URI >** (p. 1286), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1286), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1286), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1286), **decaf::util::LinkedList< Pointer< Command > >** (p. 1286), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1286), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1286), **decaf::util::LinkedList< cms::Destination * >** (p. 1286), **decaf::util::LinkedList< cms::Session * >** (p. 1286), **decaf::util::LinkedList< cms::Connection * >** (p. 1286), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2064), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 822).

6.2.3.17 template<typename E> virtual bool decaf::util::AbstractCollection< E >::tryLock ( ) [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. 2048).

6.2.3.18 template<typename E> virtual void decaf::util::AbstractCollection< E >::unlock ( ) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

6.2.3.19 template<typename E> virtual void decaf::util::AbstractCollection< E >::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.2.3.20 template<typename E> virtual void decaf::util::AbstractCollection< E >::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.2.3.21 template<typename E> virtual void decaf::util::AbstractCollection< E >::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

6.2.4 Field Documentation

6.2.4.1 template<typename E> util::concurrent::Mutex decaf::util::AbstractCollection< E >::mutex [mutable, protected]

Referenced by decaf::util::AbstractCollection< cms::Connection * >::lock(), decaf::util::AbstractCollection< cms::Connection * >::notify(), decaf::util::AbstractCollection< cms::Connection * >::notifyAll(), decaf::util::AbstractCollection< cms::Connection * >::tryLock(), decaf::util::AbstractCollection< cms::Connection * >::unlock(), and decaf::util::AbstractCollection< cms::Connection * >::wait().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractCollection.h**

6.3 decaf::util::concurrent::AbstractExecutorService Class Reference

Provides a default implementation for the methods of the **ExecutorService** (p. 1008) interface.

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

Inheritance diagram for decaf::util::concurrent::AbstractExecutorService:

Public Member Functions

- **AbstractExecutorService** ()
- virtual ~**AbstractExecutorService** ()

6.3.1 Detailed Description

Provides a default implementation for the methods of the **ExecutorService** (p. 1008) interface.

Use this class as a starting point for implementations of custom executor service implementations.

Since

1.0

6.3.2 Constructor & Destructor Documentation

6.3.2.1 decaf::util::concurrent::AbstractExecutorService::AbstractExecutorService ( )

6.3.2.2 virtual decaf::util::concurrent::AbstractExecutorService::~AbstractExecutorService ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**AbstractExecutorService.h**

6.4 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 1286) 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 >:

Data Structures

- class **ConstSimpleListIterator**
- class **SimpleListIterator**

Public Member Functions

- **AbstractList** ()
- virtual ~**AbstractList** ()
- virtual **Iterator**< E > * **iterator** ()
- virtual **Iterator**< E > * **iterator** () **const**
- virtual **ListIterator**< E > * **listIterator** ()
- virtual **ListIterator**< E > * **listIterator** () **const**
- virtual **ListIterator**< E > * **listIterator** (int index)
- virtual **ListIterator**< E > * **listIterator** (int index) **const**
- virtual void **clear** ()

  Removes all of the elements from this collection (optional operation).
- virtual bool **add** (**const** E &value)

  Returns true if this collection changed as a result of the call.
- virtual void **add** (int index **DECAF_UNUSED**, **const** E &element **DECAF_UNUSED**)
- virtual bool **addAll** (int index, **const Collection**< E > &source)

  Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual E **removeAt** (int index **DECAF_UNUSED**)

  Removes the element at the specified position in this list.
- virtual E **set** (int index **DECAF_UNUSED**, **const** E &element **DECAF_UNUSED**)
- virtual int **indexOf** (**const** E &value) **const**

  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 int **lastIndexOf** (**const** E &value) **const**

  Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

Protected Member Functions

- void **removeRange** (int start, int end)

Protected Attributes

- int **modCount**

6.4.1 Detailed Description

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

This class provides a skeletal implementation of the **List** (p. 1286) 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. 111) 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. 669) 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. 660) 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

6.4.2 Constructor & Destructor Documentation

6.4.2.1 template<typename E> decaf::util::AbstractList< E >::AbstractList ( ) [inline]

6.4.2.2 template<typename E> virtual decaf::util::AbstractList< E >::~AbstractList ( ) [inline, virtual]

6.4.3 Member Function Documentation

6.4.3.1 template<typename E> virtual bool decaf::util::AbstractList< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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. 662).

Reimplemented in **decaf::util::StlList< E >** (p. 1970), **decaf::util::ArrayList< E >** (p. 348), **decaf::util::LinkedList< E >** (p. 1271), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1271), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1271), **decaf::util::LinkedList< CompositeTask * >** (p. 1271), **decaf::util::LinkedList< URI >** (p. 1271), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1271), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1271), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1271), **decaf::util::LinkedList< Pointer< Command > >** (p. 1271), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1271), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1271), **decaf::util::LinkedList< cms::Destination * >** (p. 1271), **decaf::util::LinkedList< cms::Session * >** (p. 1271), and **decaf::util::LinkedList< cms::Connection * >** (p. 1271).

Referenced by decaf::util::AbstractList< cms::Connection * >::add(), and decaf::util::AbstractList< cms::Connection * >::addAll().

6.4.3.2 template<typename E> virtual void decaf::util::AbstractList< E >::add (int index *DECAF_UNUSED*, const E &element *DECAF_UNUSED*) [inline, virtual]

6.4.3.3 template<typename E> virtual bool decaf::util::AbstractList< E >::addAll (int *index*, const Collection< E > & *source*) [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. 660) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List**< **E** > (p. 1288).

Reimplemented in **decaf::util::StlList**< **E** > (p. 1971), **decaf::util::ArrayList**< **E** > (p. 349), **decaf::util::LinkedList**< **E** > (p. 1273), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1273), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1273), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1273), **decaf::util::LinkedList**< **URI** > (p. 1273), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1273), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1273), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1273), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1273), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1273), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1273), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1273), **decaf::util::LinkedList**< **cms::Session** * > (p. 1273), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1273), **decaf::util::AbstractSequentialList**< **E** > (p. 114), **decaf::util::AbstractSequentialList**< **Pointer**< **Transport** > > (p. 114), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** * > (p. 114), **decaf::util::AbstractSequentialList**< **CompositeTask** * > (p. 114), **decaf::util::AbstractSequentialList**< **URI** > (p. 114), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 114), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 114), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 114), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 114), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 114), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** * > (p. 114), **decaf::util::AbstractSequentialList**< **cms::Destination** * > (p. 114), **decaf::util::AbstractSequentialList**< **cms::Session** * > (p. 114), and **decaf::util::AbstractSequentialList**< **cms::Connection** * > (p. 114).

6.4.3.4 template<typename E> virtual void decaf::util::AbstractList< E >::clear ( ) [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. 1210) 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. 88).

Reimplemented in **decaf::util::StlList**< **E** > (p. 1972), **decaf::util::LinkedList**< **E** > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1274), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1274), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1274), **decaf::util::LinkedList**< **URI** > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1274), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1274), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1274), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1274), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1274), **decaf::util::LinkedList**< **cms::Session** * > (p. 1274), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1274), and **decaf::util::ArrayList**< **E** > (p. 350).

6.4.3.5 template<typename E> virtual int decaf::util::AbstractList< E >::indexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implements **decaf::util::List< E >** (p. 1290).

Reimplemented in **decaf::util::StlList< E >** (p. 1974), **decaf::util::ArrayList< E >** (p. 352), **decaf::util::LinkedList< E >** (p. 1277), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1277), **decaf::util::LinkedList< CompositeTask * >** (p. 1277), **decaf::util::LinkedList< URI >** (p. 1277), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1277), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1277), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1277), **decaf::util::LinkedList< Pointer< Command > >** (p. 1277), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1277), **decaf::util::LinkedList< cms::Destination * >** (p. 1277), **decaf::util::LinkedList< cms::Session * >** (p. 1277), and **decaf::util::LinkedList< cms::Connection * >** (p. 1277).

6.4.3.6 template<typename E> virtual Iterator<E>* decaf::util::AbstractList< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1207).

Reimplemented in **decaf::util::StlList< E >** (p. 1974), **decaf::util::AbstractSequentialList< E >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 116), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 116), **decaf::util::AbstractSequentialList< URI >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 116), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 116), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 116).

Referenced by decaf::util::ArrayList< E >::ArrayList().

6.4.3.7 template<typename E> virtual Iterator<E>* decaf::util::AbstractList< E >::iterator ( ) const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1208).

Reimplemented in **decaf::util::StlList< E >** (p. 1974), **decaf::util::AbstractSequentialList< E >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 116), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 116), **decaf::util::AbstractSequentialList< URI >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 116), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >**

(p. 116), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 116), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 116).

6.4.3.8 template<typename E> virtual int decaf::util::AbstractList< E >::lastIndexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implements **decaf::util::List< E >** (p. 1290).

Reimplemented in **decaf::util::StlList< E >** (p. 1975), **decaf::util::ArrayList< E >** (p. 352), **decaf::util::LinkedList< E >** (p. 1278), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1278), **decaf::util::LinkedList< CompositeTask * >** (p. 1278), **decaf::util::LinkedList< URI >** (p. 1278), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1278), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1278), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1278), **decaf::util::LinkedList< Pointer< Command > >** (p. 1278), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1278), **decaf::util::LinkedList< cms::Destination * >** (p. 1278), **decaf::util::LinkedList< cms::Session * >** (p. 1278), and **decaf::util::LinkedList< cms::Connection * >** (p. 1278).

6.4.3.9 template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E >::listIterator ( ) [inline, virtual]

Returns

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 1291).

Reimplemented in **decaf::util::StlList< E >** (p. 1975), **decaf::util::AbstractSequentialList< E >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 116), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 116), **decaf::util::AbstractSequentialList< URI >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 116), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 116), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 116).

Referenced by decaf::util::AbstractList< cms::Connection * >::indexOf(), decaf::util::AbstractList< cms::Connection * >::lastIndexOf(), and decaf::util::AbstractList< cms::Connection * >::removeRange().

6.4.3.10 template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E >::listIterator ( ) const [inline, virtual]

Implements **decaf::util::List< E >** (p. 1291).

Reimplemented in **decaf::util::StlList< E >** (p. 1975), **decaf::util::AbstractSequentialList< E >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 116), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 116), **decaf::util::AbstractSequentialList< URI >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 116), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 116), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 116).

6.4.3.11 template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E >::listIterator (int *index*) [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

IndexOutOfBounds-Exception	if the index is out of range (index < 0 || index > **size()** (p. 669))

Implements **decaf::util::List< E >** (p. 1292).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1278), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1278), **decaf::util::LinkedList< CompositeTask * >** (p. 1278), **decaf::util::LinkedList< URI >** (p. 1278), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1278), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1278), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1278), **decaf::util::LinkedList< Pointer< Command > >** (p. 1278), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1278), **decaf::util::LinkedList< cms::Destination * >** (p. 1278), **decaf::util::LinkedList< cms::Session * >** (p. 1278), **decaf::util::LinkedList< cms::Connection * >** (p. 1278), **decaf::util::StlList< E >** (p. 1975), **decaf::util::AbstractSequentialList< E >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 116), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 116), **decaf::util::AbstractSequentialList< URI >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 116), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 116), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 116).

6.4.3.12 template<typename E> virtual ListIterator<E>* decaf::util::AbstractList< E >::listIterator (int index) const [inline, virtual]

Implements **decaf::util::List< E >** (p. 1293).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1279), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1279), **decaf::util::LinkedList< CompositeTask * >** (p. 1279), **decaf::util::LinkedList< URI >** (p. 1279), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1279), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1279), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1279), **decaf::util::LinkedList< Pointer< Command > >** (p. 1279), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1279), **decaf::util::LinkedList< cms::Destination * >** (p. 1279), **decaf::util::LinkedList< cms::Session * >** (p. 1279), **decaf::util::LinkedList< cms::Connection * >** (p. 1279), **decaf::util::StlList< E >** (p. 1976), **decaf::util::AbstractSequentialList< E >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 117), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 117), **decaf::util::AbstractSequentialList< URI >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 117), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 117), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 117).

6.4.3.13 template<typename E> virtual E decaf::util::AbstractList< E >::removeAt (int index *index*) [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

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

Implements **decaf::util::List< E >** (p. 1293).

Reimplemented in **decaf::util::StlList< E >** (p. 1976), **decaf::util::ArrayList< E >** (p. 353), **decaf::util::AbstractSequentialList< E >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 117), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 117), **decaf::util::AbstractSequentialList< URI >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 117), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 117), and **decaf::util::AbstractSequentialList<**

cms::Connection * > (p. 117).

6.4.3.14 template<typename E> void decaf::util::AbstractList< E >::removeRange (int *start*, int *end*) [inline, protected]

Referenced by decaf::util::AbstractList< cms::Connection * >::clear().

6.4.3.15 template<typename E> virtual E decaf::util::AbstractList< E >::set (int index *DECAF_UNUSED*, const E &element *DECAF_UNUSED*) [inline, virtual]

6.4.4 Field Documentation

6.4.4.1 template<typename E> int decaf::util::AbstractList< E >::modCount [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractList.h**

6.5 decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference

This class provides a skeletal implementation of the **Map** (p. 1371) 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** ()

6.5.1 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. 1371) 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. 118). 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. 1371) 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

6.5.2 Constructor & Destructor Documentation

6.5.2.1 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**

6.6 decaf::util::concurrent::locks::AbstractOwnableSynchronizer Class Reference

Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

`#include <src/main/decaf/util/concurrent/locks/AbstractOwnableSynchronizer.h>`

Public Member Functions

- virtual ~**AbstractOwnableSynchronizer** ()

Protected Member Functions

- **AbstractOwnableSynchronizer** ()
- **decaf::lang::Thread** * **getExclusiveOwnerThread** () **const**

  Gets the Thread that was last set using the setExclusiveOwnerThread method, or NULL if no Thread has been made the exclusive owner.
- void **setExclusiveOwnerThread** (**decaf::lang::Thread** *thread)

  Sets the Thread that has exclusive ownership of this Synchronizer, can be NULL to indicate that no Thread now owns this Synchronizer.

6.6.1 Detailed Description

Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

Since

1.0

6.6.2 Constructor & Destructor Documentation

6.6.2.1 virtual decaf::util::concurrent::locks::AbstractOwnableSynchronizer::~AbstractOwnableSynchronizer ( ) [virtual]

6.6.2.2 decaf::util::concurrent::locks::AbstractOwnableSynchronizer::AbstractOwnableSynchronizer ( ) [protected]

6.6.3 Member Function Documentation

6.6.3.1 decaf::lang::Thread* decaf::util::concurrent::locks::AbstractOwnableSynchronizer::getExclusiveOwnerThread ( ) const [protected]

Gets the Thread that was last set using the setExclusiveOwnerThread method, or NULL if no Thread has been made the exclusive owner.

Returns

pointer to the owner Thread or NULL if not set.

6.6.3.2 void decaf::util::concurrent::locks::AbstractOwnableSynchronizer::setExclusiveOwnerThread (decaf::lang::Thread * *thread*) [protected]

Sets the Thread that has exclusive ownership of this Synchronizer, can be NULL to indicate that no Thread now owns this Synchronizer.

Parameters

thread	The Thread that now has ownership, or NULL if ownership is released.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**AbstractOwnableSynchronizer.h**

6.7 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 1723) 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)

  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. 660) 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	*The reference to the element to add to this* ***Collection*** *(p. 660).*

Returns

*true if the element was added to this **Collection** (p. 660).*

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

- virtual bool **addAll** (**const Collection**< E > &collection)

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	*The **Collection** (p. 660) whose elements are added to this one.*

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
IllegalArgumentException	*if some property of an element prevents it from being added to this collection*
IllegalStateException	*if an element cannot be added at this time due to insertion restrictions.*

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).

- virtual E **remove** ()

Gets and removes the element in the head of the queue.
*Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.*

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1537)	*if there is no element in the queue.*

- virtual E **element** () **const**

Gets but not removes the element in the head of the queue.
*Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.*

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1537)	*if there is no element in the queue.*

- virtual void **clear** ()

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. 1210) 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

UnsupportedOperation-Exception	*if the clear operation is not supported by this collection*

6.7.1 Detailed Description

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

This class provides skeletal implementations of some **Queue** (p. 1723) operations.

Methods add, remove, and element are based on offer, poll, and peek, respectively.

A **Queue** (p. 1723) implementation that extends this class must minimally define a method **Queue** (p. 1723). offer(E) which does not permit insertion of null elements, along with methods **Queue** (p. 1723). **peek()** (p. 1725), **Queue.-poll()** (p. 1725), **Collection.size()** (p. 669), and a **Collection.iterator()** (p. 1207) supporting **Iterator.remove()** (p. 1210). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 84).

Since

1.0

6.7.2 Constructor & Destructor Documentation

6.7.2.1 template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue ( ) [inline]

6.7.2.2 template<typename E > virtual decaf::util::AbstractQueue< E >::~AbstractQueue ( ) [inline, virtual]

6.7.3 Member Function Documentation

6.7.3.1 template<typename E > virtual bool decaf::util::AbstractQueue< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation returns true if offer succeeds, else throws an IllegalStateException.

Implements **decaf::util::Collection**< **E** > (p. 662).

Reimplemented in **decaf::util::PriorityQueue**< **E** > (p. 1678).

References decaf::util::Queue< E >::offer().

6.7.3.2 template<typename E > virtual bool decaf::util::AbstractQueue< E >::addAll (const Collection< E > & *collection*) [inline, 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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

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).

This implementation checks to see if the **Queue** (p. 1723) is being added to itself and throws an IllegalArgument-Exception if so, otherwise it delegates the add to the **AbstractCollection** (p. 84)'s addAll implementation.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 87).

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::operator=().

6.7.3.3 template<typename E > virtual void decaf::util::AbstractQueue< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

This implementation repeatedly invokes poll until it returns false.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 88).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 1679), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2059), and **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1260).

References decaf::util::AbstractCollection< E >::isEmpty(), and decaf::util::Queue< E >::poll().

6.7.3.4 template<typename E > virtual E decaf::util::AbstractQueue< E >::element ( ) const [inline, virtual]

Gets but not removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException *(p. 1537)*	if there is no element in the queue.

This implementation returns the result of peek unless the queue is empty otherwise it throws a **NoSuchElementException** (p. 1537).

Implements **decaf::util::Queue< E >** (p. 1724).

References decaf::util::Queue< E >::peek().

Referenced by decaf::util::concurrent::SynchronousQueue< E >::drainTo().

6.7.3.5 template<typename E > virtual E decaf::util::AbstractQueue< E >::remove ( ) [inline, virtual]

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException *(p. 1537)*	if there is no element in the queue.

This implementation returns the result of poll unless the queue is empty.

Implements **decaf::util::Queue< E >** (p. 1725).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 1681).

References decaf::util::Queue< E >::poll().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractQueue.h**

6.8 decaf::util::AbstractSequentialList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 1286) 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** ()
- virtual **Iterator**< E > * **iterator** ()
- virtual **Iterator**< E > * **iterator** () **const**
- virtual **ListIterator**< E > * **listIterator** ()
- virtual **ListIterator**< E > * **listIterator** () **const**
- virtual **ListIterator**< E > * **listIterator** (int index **DECAF_UNUSED**)
- virtual **ListIterator**< E > * **listIterator** (int index **DECAF_UNUSED**) **const**
- virtual E **get** (int index) **const**

  Gets the element contained at position passed.

  Parameters

index	*The position to get.*

  Returns

  value at index specified.

  Exceptions

IndexOutOfBoundsException	*if the index given is less than zero or greater than the **List** (p. 1286) size.*

- virtual E **set** (int index, **const** E &element)

  Replaces the element at the specified position in this list with the specified element.

  Parameters

index	*The index of the element to replace.*
element	*The element to be stored at the specified position.*

  Returns

  the element previously at the specified position.

  Exceptions

IndexOutOfBoundsException	*if the index given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

- virtual void **add** (int index, **const** E &element)

  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	*The index at which the specified element is to be inserted.*
element	*The element to be inserted in this **List** (p. 1286).*

Exceptions

IndexOutOfBoundsException	*if the index is greater than size of the **List** (p. 1286).*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

- virtual bool **addAll** (int index, **const Collection**< E > &source)

  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. 660) 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 given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

- virtual E **removeAt** (int index)

  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 given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*

6.8.1 Detailed Description

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

This class provides a skeletal implementation of the **List** (p. 1286) 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. 97) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p. 97) 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 listIterator 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. 660) interface specification.

Since

1.0

6.8.2 Constructor & Destructor Documentation

6.8.2.1 template<typename E> virtual decaf::util::AbstractSequentialList< E >::~AbstractSequentialList ( ) [inline, virtual]

6.8.3 Member Function Documentation

6.8.3.1 template<typename E> virtual void decaf::util::AbstractSequentialList< E >::add (int index , const E & *element*) [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	The index at which the specified element is to be inserted.
element	The element to be inserted in this **List** (p. 1286).

Exceptions

IndexOutOfBounds-Exception	if the index is greater than size of the **List** (p. 1286).
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it inserts the specified element with **ListIterator.add** (p. 1296).

Implements **decaf::util::List< E >** (p. 1287).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1272), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1272), **decaf::util::LinkedList< CompositeTask * >** (p. 1272), **decaf::util::LinkedList< URI >** (p. 1272), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1272), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1272), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1272), **decaf::util::LinkedList< Pointer< Command > >** (p. 1272), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1272), **decaf::util::LinkedList< cms::Destination * >** (p. 1272), **decaf::util::LinkedList< cms::Session * >** (p. 1272), and **decaf::util::LinkedList< cms::Connection * >** (p. 1272).

6.8.3.2 template<typename E> virtual bool decaf::util::AbstractSequentialList< E >::addAll (int *index,* const Collection< E > & *source*) [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. 660) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with listIterator(index)). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using **ListIterator.add** (p. 1296) (to skip over the added element).

Reimplemented from **decaf::util::AbstractList< E >** (p. 99).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1273), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1273), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1273), **decaf::util::LinkedList< CompositeTask * >** (p. 1273), **decaf::util::LinkedList< URI >** (p. 1273), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1273), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1273), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1273), **decaf::util::LinkedList< Pointer< Command > >** (p. 1273), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1273), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1273), **decaf::util::LinkedList< cms::Destination * >** (p. 1273), **decaf::util::LinkedList< cms::Session * >** (p. 1273), and **decaf::util::LinkedList< cms::Connection * >** (p. 1273).

6.8.3.3 template<typename E> virtual E decaf::util::AbstractSequentialList< E >::get (int *index*) const [inline, virtual]

Gets the element contained at position passed.

Parameters

index	The position to get.

Returns

value at index specified.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the element using **ListIterator.next** (p. 1209) and returns it.

Implements **decaf::util::List< E >** (p. 1289).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1276), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1276), **decaf::util::LinkedList< CompositeTask * >** (p. 1276), **decaf::util::LinkedList< URI >** (p. 1276), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1276), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1276), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1276), **decaf::util::LinkedList< Pointer< Command > >** (p. 1276), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1276), **decaf::util::LinkedList< cms::Destination * >** (p. 1276), **decaf::util::LinkedList< cms::Session * >** (p. 1276), and **decaf::util::LinkedList< cms::Connection * >** (p. 1276).

6.8.3.4 template<typename E> virtual Iterator<E>* decaf::util::AbstractSequentialList< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Reimplemented from **decaf::util::AbstractList< E >** (p. 101).

Referenced by decaf::util::LinkedList< cms::Connection * >::removeFirstOccurrence().

6.8.3.5 template<typename E> virtual Iterator<E>* decaf::util::AbstractSequentialList< E >::iterator ( ) const [inline, virtual]

Reimplemented from **decaf::util::AbstractList< E >** (p. 101).

6.8.3.6 template<typename E> virtual ListIterator<E>* decaf::util::AbstractSequentialList< E >::listIterator ( ) [inline, virtual]

Returns

a list iterator over the elements in this list (in proper sequence).

Reimplemented from **decaf::util::AbstractList< E >** (p. 102).

Referenced by decaf::util::AbstractSequentialList< cms::Connection * >::add(), decaf::util::AbstractSequentialList< cms::Connection * >::addAll(), decaf::util::AbstractSequentialList< cms::Connection * >::get(), decaf::util::AbstractSequentialList< cms::Connection * >::iterator(), decaf::util::AbstractSequentialList< cms::Connection * >::listIterator(), decaf::util::AbstractSequentialList< cms::Connection * >::removeAt(), and decaf::util::AbstractSequentialList< cms::Connection * >::set().

6.8.3.7 template<typename E> virtual ListIterator<E>* decaf::util::AbstractSequentialList< E >::listIterator ( ) const [inline, virtual]

Reimplemented from **decaf::util::AbstractList< E >** (p. 103).

6.8.3.8 template<typename E> virtual ListIterator<E>* decaf::util::AbstractSequentialList< E >::listIterator (int index *index*) [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

IndexOutOfBounds-Exception	if the index is out of range (index < 0 || index > **size()** (p. 669))

Reimplemented from **decaf::util::AbstractList< E >** (p. 103).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1278), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1278), **decaf::util::LinkedList< CompositeTask * >** (p. 1278), **decaf::util::LinkedList< URI >** (p. 1278), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1278), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1278), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1278), **decaf::util::LinkedList< Pointer< Command > >** (p. 1278), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1278), **decaf::util::LinkedList< cms::Destination * >** (p. 1278), **decaf::util::LinkedList< cms::Session * >** (p. 1278), and **decaf::util::LinkedList< cms::Connection * >** (p. 1278).

6.8.3.9 template<typename E> virtual ListIterator<E>* decaf::util::AbstractSequentialList< E >::listIterator (int index *DECAF_UNUSED*) const [inline, virtual]

Reimplemented from **decaf::util::AbstractList< E >** (p. 103).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1279), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1279), **decaf::util::LinkedList< CompositeTask * >** (p. 1279), **decaf::util::LinkedList< URI >** (p. 1279), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1279), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1279), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1279), **decaf::util::LinkedList< Pointer< Command > >** (p. 1279), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1279), **decaf::util::LinkedList< cms::Destination * >** (p. 1279), **decaf::util::LinkedList< cms::Session * >** (p. 1279), and **decaf::util::LinkedList< cms::Connection * >** (p. 1279).

6.8.3.10 template<typename E> virtual E decaf::util::AbstractSequentialList< E >::removeAt (int index) [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

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it removes the element with **ListIterator.remove** (p. 1210).

Reimplemented from **decaf::util::AbstractList< E >** (p. 104).

6.8.3.11 template<typename E> virtual E decaf::util::AbstractSequentialList< E >::set (int index , const E & *element*) [inline, virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters

index	The index of the element to replace.
element	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the current element using **ListIterator.next** (p. 1209) and replaces it with **ListIterator.set** (p. 1297).

Implements **decaf::util::List< E >** (p. 1294).

Reimplemented in **decaf::util::LinkedList< E >** (p. 1285), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1285), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1285), **decaf::util::LinkedList< CompositeTask * >** (p. 1285), **decaf::util::LinkedList< URI >** (p. 1285), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1285), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1285), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1285), **decaf::util::LinkedList< Pointer< Command > >** (p. 1285), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1285), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1285), **decaf::util::LinkedList< cms::Destination * >** (p. 1285), **decaf::util::LinkedList< cms::Session * >** (p. 1285), and **decaf::util::LinkedList< cms::Connection * >** (p. 1285).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractSequentialList.h**

6.9 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 1857) 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)

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. 660) whose elements are to be removed from this one.*

Returns

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

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

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.

6.9.1 Detailed Description

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

This class provides a skeletal implementation of the **Set** (p. 1857) 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. 660) by extending **AbstractCollection** (p. 84), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 1857) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since

1.0

6.9.2 Constructor & Destructor Documentation

6.9.2.1 template<typename E> virtual decaf::util::AbstractSet< E >::~AbstractSet ( ) [inline, virtual]

6.9.3 Member Function Documentation

6.9.3.1 template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll (const Collection< E > & *collection*) [inline, 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. 660) whose elements are to be removed from this one.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 93).

Reimplemented in **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 821).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractSet.h**

6.10 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the **TransportFactory** (p. 2167) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2167) 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)

  *Creates the WireFormat that is configured for this **Transport** (p. 2161) and returns it.*

6.10.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 2167) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2167) instances.

Since

3.0

6.10.2 Constructor & Destructor Documentation

6.10.2.1 virtual activemq::transport::AbstractTransportFactory::~AbstractTransportFactory ( ) [inline, virtual]

6.10.3 Member Function Documentation

6.10.3.1 virtual Pointer<wireformat::WireFormat> activemq::transport::AbstractTransportFactory::createWireFormat (const decaf::util::Properties & *properties*) [protected, virtual]

Creates the WireFormat that is configured for this **Transport** (p. 2161) 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**

6.11 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

  Method called to acknowledge the message once it has been received by a MessageConsumer.

6.11.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

Since

2.0

6.11.2 Constructor & Destructor Documentation

6.11.2.1 virtual activemq::core::ActiveMQAckHandler::~ActiveMQAckHandler ( ) [inline, virtual]

6.11.3 Member Function Documentation

6.11.3.1 virtual void activemq::core::ActiveMQAckHandler::acknowledgeMessage (const commands::Message * *message*) [pure virtual]

Method called to acknowledge the message once it has been received by a MessageConsumer.

Parameters

message	The Message to Acknowledge.

Exceptions

CMSException	if an error occurs while acknowledging the given Message.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQAckHandler.h**

6.12 activemq::commands::ActiveMQBlobMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQBlobMessage.h>`

Inheritance diagram for activemq::commands::ActiveMQBlobMessage:

Public Member Functions

- **ActiveMQBlobMessage** ()
- virtual ~**ActiveMQBlobMessage** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure*** (p. *877*) *Type as defined in CommandTypes.h.*
- virtual **ActiveMQBlobMessage** * **cloneDataStructure** () **const**

  Clone this obbject 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. *877*) *such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure*** (p. *877*) *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**

6.12.1 Constructor & Destructor Documentation

6.12.1.1 activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ( )

6.12.1.2 virtual activemq::commands::ActiveMQBlobMessage::~ActiveMQBlobMessage ( ) throw () [inline, virtual]

6.12.2 Member Function Documentation

6.12.2.1 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

Implements **cms::Message** (p. 1430).

6.12.2.2 virtual ActiveMQBlobMessage* activemq::commands::ActiveMQBlobMessage::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1417).

6.12.2.3 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. 1417).

6.12.2.4 virtual bool activemq::commands::ActiveMQBlobMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230).

6.12.2.5 virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1419).

6.12.2.6 std::string activemq::commands::ActiveMQBlobMessage::getMimeType ( ) const [inline]

Get the Mime Type of the Blob.

Returns

string holding the MIME Type.

6.12.2.7 std::string activemq::commands::ActiveMQBlobMessage::getName ( ) const [inline]

Gets the Name of the Blob.

Returns

string name of the Blob.

6.12.2.8 std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl ( ) const [inline]

Get the Remote URL of the Blob.

Returns

string from of the Remote Blob URL.

6.12.2.9 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.

6.12.2.10 void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker (bool *value*) [inline]

Sets the Deleted By Broker flag.

Parameters

value	- set the Delete by broker flag to value.

6.12.2.11 void activemq::commands::ActiveMQBlobMessage::setMimeType (const std::string & *mimeType*) [inline]

Set the Mime Type of the Blob.

Parameters

mimeType	- String holding the MIME Type.

6.12.2.12 void activemq::commands::ActiveMQBlobMessage::setName (const std::string & *name*) [inline]

Sets the Name of the Blob.

Parameters

name	- Name of the Blob.

6.12.2.13 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.

6.12.2.14 virtual std::string activemq::commands::ActiveMQBlobMessage::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1424).

6.12.3 Field Documentation

6.12.3.1 const std::string activemq::commands::ActiveMQBlobMessage::BINARY_MIME_TYPE [static]

6.12.3.2 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**

6.13 activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessage-Marshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 126).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ-BlobMessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual ~**ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.13.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 126).

NOTE!: This file is 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::generated::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller ( ) [inline]

6.13.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::~ActiveMQBlobMessageMarshaller ( ) [inline, virtual]

6.13.3 Member Function Documentation

6.13.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.13.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.13.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.13.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.13.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.13.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.13.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQBlobMessageMarshaller.h**

6.14 activemq::commands::ActiveMQBytesMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQBytesMessage.h>`

Inheritance diagram for activemq::commands::ActiveMQBytesMessage:

Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual ∼**ActiveMQBytesMessage** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQBytesMessage** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage** ∗ **clone** () **const**

  Clones this message.
- virtual void **clearBody** ()

  Clears out the body of the message.
- virtual void **onSend** ()

  *Allows derived **Message** (p. 1412) classes to perform tasks before a message is sent.*
- virtual void **setBodyBytes** (**const** unsigned char ∗buffer, int numBytes)

  sets the bytes given to the message body.
- virtual unsigned char ∗ **getBodyBytes** () **const**

  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 int **getBodyLength** () **const**

  Returns the number of bytes contained in the body of this message.
- virtual void **reset** ()

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

- virtual bool **readBoolean** () **const**

  Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value)

  Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () **const**

  Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value)

  Writes a byte to the bytes message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) **const**

  Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (**const** std::vector< unsigned char > &value)

  Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) **const**

  Reads a portion of the bytes message stream.
- virtual void **writeBytes** (**const** unsigned char *value, int offset, int length)

  Writes a portion of a byte array to the bytes message stream.
- virtual char **readChar** () **const**

  Reads a Char from the Bytes message stream.
- virtual void **writeChar** (char value)

  Writes a char to the bytes message stream as a 1-byte value.
- virtual float **readFloat** () **const**

  Reads a 32 bit float from the Bytes message stream.
- virtual void **writeFloat** (float value)

  Writes a float to the bytes message stream as a 4 byte value.
- virtual double **readDouble** () **const**

  Reads a 64 bit double from the Bytes message stream.
- virtual void **writeDouble** (double value)

  Writes a double to the bytes message stream as a 8 byte value.
- virtual short **readShort** () **const**

  Reads a 16 bit signed short from the Bytes message stream.
- virtual void **writeShort** (short value)

  Writes a signed short to the bytes message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () **const**

  Reads a 16 bit unsigned short from the Bytes message stream.
- virtual void **writeUnsignedShort** (unsigned short value)

  Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual int **readInt** () **const**

  Reads a 32 bit signed integer from the Bytes message stream.
- virtual void **writeInt** (int value)

  Writes a signed int to the bytes message stream as a 4 byte value.
- virtual long long **readLong** () **const**

  Reads a 64 bit long from the Bytes message stream.
- virtual void **writeLong** (long long value)

  Writes a long long to the bytes message stream as a 8 byte value.
- virtual std::string **readString** () **const**

  Reads an ASCII String from the Bytes message stream.
- virtual void **writeString** (**const** std::string &value)

  Writes an ASCII String to the Bytes message stream.
- virtual std::string **readUTF** () **const**

  Reads an UTF String from the BytesMessage stream.
- virtual void **writeUTF** (**const** std::string &value)

  Writes an UTF String to the BytesMessage stream.

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQBYTESMESSAGE** = 24

6.14.1 Constructor & Destructor Documentation

6.14.1.1 activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ( )

6.14.1.2 virtual activemq::commands::ActiveMQBytesMessage::~ActiveMQBytesMessage ( ) throw () [virtual]

6.14.2 Member Function Documentation

6.14.2.1 virtual void activemq::commands::ActiveMQBytesMessage::clearBody ( ) [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

CMSException	- if an internal error occurs.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230).

6.14.2.2 virtual cms::BytesMessage* activemq::commands::ActiveMQBytesMessage::clone ( ) const [inline, virtual]

Clones this message.

Returns

a deep copy of this message.

Exceptions

CMSException	- if an internal error occurs while cloning the **Message** (p. 1412).

Implements **cms::BytesMessage** (p. 559).

6.14.2.3 virtual ActiveMQBytesMessage* activemq::commands::ActiveMQBytesMessage::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1417).

6.14.2.4 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. 1417).

6.14.2.5 virtual bool activemq::commands::ActiveMQBytesMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230).

6.14.2.6 virtual unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes ( ) const [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	- If an internal error occurs.
MessageNotReadable-Exception	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 559).

6.14.2.7 virtual int activemq::commands::ActiveMQBytesMessage::getBodyLength ( ) const [virtual]

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

CMSException	- If an internal error occurs.
MessageNotReadable-Exception	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 559).

6.14.2.8 virtual unsigned char activemq::commands::ActiveMQBytesMessage::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1419).

6.14.2.9 virtual void activemq::commands::ActiveMQBytesMessage::onSend ( ) [virtual]

Allows derived **Message** (p. 1412) classes to perform tasks before a message is sent.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232).

6.14.2.10 virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 560).

6.14.2.11 virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 560).

6.14.2.12 virtual int activemq::commands::ActiveMQBytesMessage::readBytes (std::vector< unsigned char > & *value*) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 560).

6.14.2.13 virtual int activemq::commands::ActiveMQBytesMessage::readBytes (unsigned char * *buffer,* int *length*) const [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

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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 561).

6.14.2.14 virtual char activemq::commands::ActiveMQBytesMessage::readChar ( ) const [virtual]

Reads a Char from the Bytes 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 bytes stream has been reached.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 562).

6.14.2.15 virtual double activemq::commands::ActiveMQBytesMessage::readDouble ( ) const [virtual]

Reads a 64 bit double 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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 562).

6.14.2.16 virtual float activemq::commands::ActiveMQBytesMessage::readFloat ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 562).

6.14.2.17 virtual int activemq::commands::ActiveMQBytesMessage::readInt ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 563).

6.14.2.18 virtual long long activemq::commands::ActiveMQBytesMessage::readLong ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 563).

6.14.2.19 virtual short activemq::commands::ActiveMQBytesMessage::readShort ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 563).

6.14.2.20 virtual std::string activemq::commands::ActiveMQBytesMessage::readString ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 564).

6.14.2.21 virtual unsigned short activemq::commands::ActiveMQBytesMessage::readUnsignedShort ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 564).

6.14.2.22 virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 564).

6.14.2.23 virtual void activemq::commands::ActiveMQBytesMessage::reset ( ) [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. 1412) has an invalid format.

Implements **cms::BytesMessage** (p. 565).

6.14.2.24 virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes (const unsigned char * *buffer*, int *numBytes*) [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.
MessageNotWriteable-Exception	- if in Read Only Mode.

Implements **cms::BytesMessage** (p. 565).

6.14.2.25 virtual std::string activemq::commands::ActiveMQBytesMessage::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1424).

6.14.2.26 virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean (bool *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 565).

6.14.2.27 virtual void activemq::commands::ActiveMQBytesMessage::writeByte (unsigned char *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 565).

6.14.2.28 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const std::vector< unsigned char > & *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 566).

6.14.2.29 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const unsigned char * *value*, int *offset*, int *length*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 566).

6.14.2.30 virtual void activemq::commands::ActiveMQBytesMessage::writeChar (char *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 566).

6.14.2.31 virtual void activemq::commands::ActiveMQBytesMessage::writeDouble (double *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 567).

6.14.2.32 virtual void activemq::commands::ActiveMQBytesMessage::writeFloat (float *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 567).

6.14.2.33 virtual void activemq::commands::ActiveMQBytesMessage::writeInt (int *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 567).

6.14.2.34 virtual void activemq::commands::ActiveMQBytesMessage::writeLong (long long *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 568).

6.14.2.35 virtual void activemq::commands::ActiveMQBytesMessage::writeShort (short *value*) [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	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 568).

6.14.2.36 virtual void activemq::commands::ActiveMQBytesMessage::writeString (const std::string & *value*) [virtual]

Writes an ASCII String to the Bytes 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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 568).

6.14.2.37 virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort (unsigned short *value*) [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	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 569).

6.14.2.38 virtual void activemq::commands::ActiveMQBytesMessage::writeUTF (const std::string & *value*) [virtual]

Writes an UTF String to the BytesMessage stream.

Parameters

value	String to write to the stream

Exceptions

CMSException	- if the CMS provider fails to read the message due to some internal error.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 569).

6.14.3 Field Documentation

6.14.3.1 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**

6.15 activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 142).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQBytesMessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.15.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 142).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.15.2 Constructor & Destructor Documentation

6.15.2.1 activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller ( ) [inline]

6.15.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::~ActiveMQBytesMessageMarshaller ( ) [inline, virtual]

6.15.3 Member Function Documentation

6.15.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.15.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.15.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.15.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.15.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.15.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.15.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQBytesMessageMarshaller.h**

6.16 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** () throw ()
- virtual void **addSession** (**ActiveMQSession** *session)

  Adds the session resources for the given session instance.
- virtual void **removeSession** (**ActiveMQSession** *session)

  Removes the session resources for the given session instance.
- virtual void **addProducer** (**ActiveMQProducer** *producer)

  Adds an active Producer to the Set of known producers.
- virtual void **removeProducer** (**const Pointer**< **commands::ProducerId** > &producerId)

  Removes an active Producer to the Set of known producers.
- virtual void **addDispatcher** (**const Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** *dispatcher)

  Adds a dispatcher for a consumer.
- virtual void **removeDispatcher** (**const Pointer**< **commands::ConsumerId** > &consumer)

  Removes the dispatcher for a consumer.
- virtual void **sendPullRequest** (**const commands::ConsumerInfo** *consumer, long long timeout)

  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.
- bool **isTransportFailed** () **const**

  Checks if the Connection's Transport has failed.
- virtual void **destroyDestination** (**const commands::ActiveMQDestination** *destination)

  Requests that the Broker removes the given Destination.
- virtual void **destroyDestination** (**const cms::Destination** *destination)

  Requests that the Broker removes the given Destination.
- virtual **const**
  cms::ConnectionMetaData * **getMetaData** () **const**

  Gets the metadata for this connection.

  Returns

  the connection MetaData pointer (caller does not own it).

  Exceptions

CMSException (p. 640)	*if the provider fails to get the connection metadata for this connection.*

See also

ConnectionMetaData *(*p. *759)*

Since

2.0

- virtual **cms::Session** ∗ **createSession** ()

  Creates an AUTO_ACKNOWLEDGE ***Session*** *(*p. *1830).*

  Exceptions

CMSException (p. 640)	

- virtual std::string **getClientID** () **const**

  Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

  Returns

  Client Id String for this ***Connection*** *(*p. *725).*

  Exceptions

CMSException (p. 640)	*if the provider fails to return the client id or an internal error occurs.*

- virtual void **setClientID** (**const** std::string &clientID)

  Sets the client identifier for this connection.
  The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific ***ConnectionFactory*** *(*p. *741) object and transparently assigned to the* ***Connection*** *(*p. *725) object it creates.*
  If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an ***IllegalStateException*** *(*p. *1098).*

  Parameters

clientID	*The unique client identifier to assign to the* ***Connection*** *(*p. *725).*

  Exceptions

CMSException (p. 640)	*if the provider fails to set the client id due to some internal error.*
InvalidClientIDException	*if the id given is somehow invalid or is a duplicate.*
IllegalStateException (p. 1098)	*if the client tries to set the id after a* ***Connection*** *(*p. *725) method has been called.*

- virtual **cms::Session** ∗ **createSession** (**cms::Session::AcknowledgeMode** ackMode)

  Creates a new ***Session*** *(*p. *1830) to work for this* ***Connection*** *(*p. *725) using the specified acknowledgment mode.*

  Parameters

ackMode	*the Acknowledgment Mode to use.*

  Exceptions

CMSException (p. 640)	

- virtual void **close** ()

  Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

  Exceptions

CMSException (p. 640)	

- virtual void **start** ()

  Starts the service.

  Exceptions

CMSException (p. 640)	*if an internal error occurs while starting.*

- virtual void **stop** ()

  Stops this service.

Exceptions

CMSException (p. 640)	*- if an internal error occurs while stopping the Service.*

- virtual **cms::ExceptionListener** ∗ **getExceptionListener** () **const**

  Gets the registered Exception Listener for this connection.

  Returns

  pointer to an exception listener or NULL

- virtual void **setExceptionListener** (**cms::ExceptionListener** ∗listener)

  Sets the registered Exception Listener for this connection.

  Parameters

listener	*pointer to and* ***ExceptionListener*** *(p. 996)*

- void **setUsername** (**const** std::string &username)

  Sets the username that should be used when creating a new connection.

- **const** std::string & **getUsername** () **const**

  Gets the username that this factory will use when creating a new connection instance.

- void **setPassword** (**const** std::string &password)

  Sets the password that should be used when creating a new connection.

- **const** std::string & **getPassword** () **const**

  Gets the password that this factory will use when creating a new connection instance.

- void **setDefaultClientId** (**const** std::string &clientId)

  Sets the Client Id.

- void **setBrokerURL** (**const** std::string &brokerURL)

  Sets the Broker URL that should be used when creating a new connection instance.

- **const** std::string & **getBrokerURL** () **const**

  Gets the Broker URL that this factory will use when creating a new connection instance.

- void **setPrefetchPolicy** (**PrefetchPolicy** ∗policy)

  Sets the ***PrefetchPolicy*** *(p. 1635) instance that this factory should use when it creates new Connection instances.*

- **PrefetchPolicy** ∗ **getPrefetchPolicy** () **const**

  Gets the pointer to the current ***PrefetchPolicy*** *(p. 1635) that is in use by this ConnectionFactory.*

- void **setRedeliveryPolicy** (**RedeliveryPolicy** ∗policy)

  Sets the ***RedeliveryPolicy*** *(p. 1744) instance that this factory should use when it creates new Connection instances.*

- **RedeliveryPolicy** ∗ **getRedeliveryPolicy** () **const**

  Gets the pointer to the current ***RedeliveryPolicy*** *(p. 1744) that is in use by this ConnectionFactory.*

- bool **isDispatchAsync** () **const**
- void **setDispatchAsync** (bool value)

  Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

- 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.

- bool **isUseCompression** () **const**

  Gets if the Connection is configured for Message body compression.

- void **setUseCompression** (bool value)

  Sets whether Message body compression is enabled.

- void **setCompressionLevel** (int value)

Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.

- int **getCompressionLevel** () **const**

  Gets the currently configured Compression level for Message bodies.
- 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.
- bool **isMessagePrioritySupported** () **const**
- void **setMessagePrioritySupported** (bool value)

  Set whether or not this factory should create Connection objects with the Message priority support function enabled.
- long long **getNextTempDestinationId** ()

  Get the Next Temporary Destination Id.
- long long **getNextLocalTransactionId** ()

  Get the Next Temporary Destination Id.
- 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**

  Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.
- **const commands::ConnectionId** & **getConnectionId** () **const**

  Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.
- **transport::Transport** & **getTransport** () **const**

  Gets a reference to this object's Transport instance.
- **Pointer**< **threads::Scheduler** > **getScheduler** () **const**

  Gets a reference to the Connection objects built in Scheduler instance.
- std::string **getResourceManagerId** () **const**

  Returns the Id of the Resource Manager that this client will use should it be entered into an XA Transaction.
- void **cleanup** ()

  *Clean up this connection object, reseting it back to a state that mirrors what a newly created **ActiveMQConnection** (p. 145) object has.*

- void **oneway** (**Pointer**< **commands::Command** > command)

  Sends a message without request that the broker send a response to indicate that it was received.
- **Pointer**< **commands::Response** > **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0)

  Sends a synchronous request and returns the response from the broker.
- virtual void **fire** (**const exceptions::ActiveMQException** &ex)

  Notify the exception listener.
- void **setTransportInterruptionProcessingComplete** ()

  Indicates that a Connection resource that is processing the transportInterrupted event has completed.
- **decaf::lang::Exception** ∗ **getFirstFailureError** () **const**

  Gets the pointer to the first exception that caused the Connection to become failed.
- void **onAsyncException** (**const decaf::lang::Exception** &ex)

  Event handler for dealing with async exceptions.
- void **checkClosed** () **const**

  Check for Closed State and Throw an exception if true.
- void **checkClosedOrFailed** () **const**

  Check for Closed State and Failed State and Throw an exception if either is true.
- void **ensureConnectionInfoSent** ()

  If its not been sent, then send the ConnectionInfo to the Broker.

Protected Member Functions

- virtual **Pointer**
  < **commands::SessionId** > **getNextSessionId** ()
- void **disconnect** (long long lastDeliveredSequenceId)
- void **waitForTransportInterruptionProcessingToComplete** ()
- void **signalInterruptionProcessingComplete** ()
- **const decaf::util::Properties** & **getProperties** () **const**

6.16.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

Since

2.0

6.16.2 Constructor & Destructor Documentation

6.16.2.1 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

6.16.2.2 virtual activemq::core::ActiveMQConnection::~ActiveMQConnection ( ) throw () [virtual]

6.16.3 Member Function Documentation

6.16.3.1 virtual void activemq::core::ActiveMQConnection::addDispatcher (const Pointer< commands::ConsumerId > & *consumer,* Dispatcher * *dispatcher*) [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.

Exceptions

CMSException	if an error occurs while removing performing the operation.

6.16.3.2 virtual void activemq::core::ActiveMQConnection::addProducer (ActiveMQProducer * *producer*) [virtual]

Adds an active Producer to the Set of known producers.

Parameters

producer	The Producer to add from the the known set.

Exceptions

CMSException	if an error occurs while removing performing the operation.

6.16.3.3 virtual void activemq::core::ActiveMQConnection::addSession (ActiveMQSession * *session*) [virtual]

Adds the session resources for the given session instance.

Parameters

session	The session to be added to this connection.

Exceptions

CMSException	if an error occurs while removing performing the operation.

6.16.3.4 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.

6.16.3.5 void activemq::core::ActiveMQConnection::checkClosed ( ) const

Check for Closed State and Throw an exception if true.

Exceptions

CMSException	if the Connection is closed.

6.16.3.6 void activemq::core::ActiveMQConnection::checkClosedOrFailed ( ) const

Check for Closed State and Failed State and Throw an exception if either is true.

Exceptions

CMSException	if the Connection is closed or failed.

6.16.3.7 void activemq::core::ActiveMQConnection::cleanup ( )

Clean up this connection object, reseting it back to a state that mirrors what a newly created **ActiveMQConnection** (p. 145) object has.

6.16.3.8 virtual void activemq::core::ActiveMQConnection::close ( ) [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

CMSException *(p. 640)*	

Implements **cms::Connection** (p. 726).

6.16.3.9 virtual cms::Session* activemq::core::ActiveMQConnection::createSession ( ) [virtual]

Creates an AUTO_ACKNOWLEDGE **Session** (p. 1830).

Exceptions

CMSException *(p. 640)*	

Implements **cms::Connection** (p. 726).

6.16.3.10 virtual cms::Session* activemq::core::ActiveMQConnection::createSession (cms::Session::AcknowledgeMode *ackMode*) [virtual]

Creates a new **Session** (p. 1830) to work for this **Connection** (p. 725) using the specified acknowledgment mode.

Parameters

ackMode	the Acknowledgment Mode to use.

Exceptions

CMSException *(p. 640)*	

Implements **cms::Connection** (p. 726).

Reimplemented in **activemq::core::ActiveMQXAConnection** (p. 335).

6.16.3.11 virtual void activemq::core::ActiveMQConnection::destroyDestination (const commands::ActiveMQDestination * *destination*) [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.
UnsupportedOperation-Exception	If the wire format in use does not support this operation.
ActiveMQException	If any other error occurs during the attempt to destroy the destination.

6.16.3.12 virtual void activemq::core::ActiveMQConnection::destroyDestination (const cms::Destination * *destination*) [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.
UnsupportedOperation-Exception	If the wire format in use does not support this operation.
ActiveMQException	If any other error occurs during the attempt to destroy the destination.

6.16.3.13 void activemq::core::ActiveMQConnection::disconnect (long long *lastDeliveredSequenceId*) [protected]

6.16.3.14 void activemq::core::ActiveMQConnection::ensureConnectionInfoSent ( )

If its not been sent, then send the ConnectionInfo to the Broker.

6.16.3.15 virtual void activemq::core::ActiveMQConnection::fire (const exceptions::ActiveMQException & *ex*) [virtual]

Notify the exception listener.

Parameters

ex	the exception to fire

6.16.3.16 const std::string& activemq::core::ActiveMQConnection::getBrokerURL ( ) const

Gets the Broker URL that this factory will use when creating a new connection instance.

Returns

brokerURL string

6.16.3.17 virtual std::string activemq::core::ActiveMQConnection::getClientID ( ) const [virtual]

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

Returns

Client Id String for this **Connection** (p. 725).

Exceptions

CMSException *(p. 640)*	if the provider fails to return the client id or an internal error occurs.

Implements **cms::Connection** (p. 727).

6.16.3.18 unsigned int activemq::core::ActiveMQConnection::getCloseTimeout ( ) const

Gets the assigned close timeout for this Connector.

Returns

the close timeout configured in the connection uri

6.16.3.19 int activemq::core::ActiveMQConnection::getCompressionLevel ( ) const

Gets the currently configured Compression level for Message bodies.

Returns

the int value of the current compression level.

6.16.3.20 const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId ( ) const

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

Exceptions

ActiveMQException	if an error occurs while performing this operation.

6.16.3.21 const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo ( ) const

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

Exceptions

ActiveMQException	if an error occurs while performing this operation.

6.16.3.22 virtual cms::ExceptionListener* activemq::core::ActiveMQConnection::getExceptionListener ( ) const [virtual]

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

Implements **cms::Connection** (p. 727).

6.16.3.23 decaf::lang::Exception* activemq::core::ActiveMQConnection::getFirstFailureError ( ) const

Gets the pointer to the first exception that caused the Connection to become failed.

Returns

pointer to and Exception instance or NULL if none is set.

6.16.3.24 virtual const cms::ConnectionMetaData* activemq::core::ActiveMQConnection::getMetaData ( ) const [inline, virtual]

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSException *(p. 640)*	if the provider fails to get the connection metadata for this connection.

See also

ConnectionMetaData (p. 759)

Since

2.0

Implements **cms::Connection** (p. 727).

6.16.3.25 long long activemq::core::ActiveMQConnection::getNextLocalTransactionId ( )

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.16.3.26 virtual Pointer<commands::SessionId> activemq::core::ActiveMQConnection::getNextSessionId ( ) [protected, virtual]

Returns

the next available Session Id.

6.16.3.27 long long activemq::core::ActiveMQConnection::getNextTempDestinationId ( )

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.16.3.28 const std::string& activemq::core::ActiveMQConnection::getPassword ( ) const

Gets the password that this factory will use when creating a new connection instance.

Returns

password string, "" for default credentials

6.16.3.29 PrefetchPolicy* activemq::core::ActiveMQConnection::getPrefetchPolicy ( ) const

Gets the pointer to the current **PrefetchPolicy** (p. 1635) that is in use by this ConnectionFactory.

Returns

a pointer to this objects **PrefetchPolicy** (p. 1635).

6.16.3.30 unsigned int activemq::core::ActiveMQConnection::getProducerWindowSize ( ) const

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.

6.16.3.31 const decaf::util::Properties& activemq::core::ActiveMQConnection::getProperties ( ) const [protected]

6.16.3.32 RedeliveryPolicy* activemq::core::ActiveMQConnection::getRedeliveryPolicy ( ) const

Gets the pointer to the current **RedeliveryPolicy** (p. 1744) that is in use by this ConnectionFactory.

Returns

a pointer to this objects **RedeliveryPolicy** (p. 1744).

6.16.3.33 std::string activemq::core::ActiveMQConnection::getResourceManagerId ( ) const

Returns the Id of the Resource Manager that this client will use should it be entered into an XA Transaction.

Returns

a string containing the resource manager Id for XA Transactions.

6.16.3.34 Pointer<threads::Scheduler> activemq::core::ActiveMQConnection::getScheduler ( ) const

Gets a reference to the Connection objects built in Scheduler instance.

Returns

a reference to a Scheduler instance owned by this Connection.

6.16.3.35 unsigned int activemq::core::ActiveMQConnection::getSendTimeout ( ) const

Gets the assigned send timeout for this Connector.

Returns

the send timeout configured in the connection uri

6.16.3.36 transport::Transport& activemq::core::ActiveMQConnection::getTransport ( ) const

Gets a reference to this object's Transport instance.

Returns

a reference to the Transport that is in use by this Connection.

6.16.3.37 const std::string& activemq::core::ActiveMQConnection::getUsername ( ) const

Gets the username that this factory will use when creating a new connection instance.

Returns

username string, "" for default credentials

6.16.3.38 bool activemq::core::ActiveMQConnection::isAlwaysSyncSend ( ) const

Gets if the Connection should always send things Synchronously.

Returns

true if sends should always be Synchronous.

6.16.3.39 bool activemq::core::ActiveMQConnection::isClosed ( ) const [inline]

Checks if this connection has been closed.

Returns

true if the connection is closed

6.16.3.40 bool activemq::core::ActiveMQConnection::isDispatchAsync ( ) const

Returns

The value of the dispatch asynchronously option sent to the broker.

6.16.3.41 bool activemq::core::ActiveMQConnection::isMessagePrioritySupported ( ) const

Returns

true if the Connections that this factory creates should support the message based priority settings.

6.16.3.42 bool activemq::core::ActiveMQConnection::isStarted ( ) const [inline]

Check if this connection has been started.

Returns

true if the start method has been called.

6.16.3.43 bool activemq::core::ActiveMQConnection::isTransportFailed ( ) const [inline]

Checks if the Connection's Transport has failed.

Returns

true if the Connection's Transport has failed.

6.16.3.44 bool activemq::core::ActiveMQConnection::isUseAsyncSend ( ) const

Gets if the useAsyncSend option is set.

Returns

true if on false if not.

6.16.3.45 bool activemq::core::ActiveMQConnection::isUseCompression ( ) const

Gets if the Connection is configured for Message body compression.

Returns

if the Message body will be Compressed or not.

6.16.3.46 void activemq::core::ActiveMQConnection::onAsyncException (const decaf::lang::Exception & *ex*)

Event handler for dealing with async exceptions.

Parameters

ex	The exception that caused the error condition.

6.16.3.47 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. 2177).

6.16.3.48 void activemq::core::ActiveMQConnection::oneway (Pointer< commands::Command > *command*)

Sends a message without request that the broker send a response to indicate that it was received.

Parameters

command	The Command object to send to the Broker.

Exceptions

ActiveMQException	if not currently connected, or if the operation fails for any reason.

6.16.3.49 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. 2177).

6.16.3.50 virtual void activemq::core::ActiveMQConnection::removeDispatcher (const Pointer< commands::ConsumerId > & *consumer*) [virtual]

Removes the dispatcher for a consumer.

Parameters

consumer	- The consumer for which to remove the dispatcher.

Exceptions

CMSException	if an error occurs while removing performing the operation.

6.16.3.51 virtual void activemq::core::ActiveMQConnection::removeProducer (const Pointer< commands::ProducerId > & *producerId*) [virtual]

Removes an active Producer to the Set of known producers.

Parameters

producerId	- The ProducerId to remove from the the known set.

Exceptions

CMSException	if an error occurs while removing performing the operation.

6.16.3.52 virtual void activemq::core::ActiveMQConnection::removeSession (ActiveMQSession * *session*) [virtual]

Removes the session resources for the given session instance.

Parameters

session	The session to be unregistered from this connection.

Exceptions

CMSException	if an error occurs while removing performing the operation.

6.16.3.53 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.

6.16.3.54 virtual void activemq::core::ActiveMQConnection::sendPullRequest (const commands::ConsumerInfo * *consumer*, long long *timeout*) [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.

Exceptions

ActiveMQException	if an error occurs while removing performing the operation.

6.16.3.55 void activemq::core::ActiveMQConnection::setAlwaysSyncSend (bool *value*)

Sets if the Connection should always send things Synchronously.

Parameters

value	true if sends should always be Synchronous.

6.16.3.56 void activemq::core::ActiveMQConnection::setBrokerURL (const std::string & *brokerURL*)

Sets the Broker URL that should be used when creating a new connection instance.

Parameters

brokerURL	string

6.16.3.57 virtual void activemq::core::ActiveMQConnection::setClientID (const std::string & *clientID*) [virtual]

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 741) object and transparently assigned to the **Connection** (p. 725) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1098).

Parameters

clientID	The unique client identifier to assign to the **Connection** (p. 725).

Exceptions

CMSException *(p. 640)*	if the provider fails to set the client id due to some internal error.
InvalidClientIDException	if the id given is somehow invalid or is a duplicate.
IllegalStateException *(p. 1098)*	if the client tries to set the id after a **Connection** (p. 725) method has been called.

Implements **cms::Connection** (p. 727).

6.16.3.58 void activemq::core::ActiveMQConnection::setCloseTimeout (unsigned int *timeout*)

Sets the close timeout to use when sending the disconnect request.

Parameters

timeout	- The time to wait for a close message.

6.16.3.59 void activemq::core::ActiveMQConnection::setCompressionLevel (int *value*)

Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.

The range of compression levels is [0..9] where 0 indicates best speed and 9 indicates best compression.

Parameters

value	A signed int value that controls the compression level.

6.16.3.60 void activemq::core::ActiveMQConnection::setDefaultClientId (const std::string & *clientId*)

Sets the Client Id.

Parameters

clientId	- The new clientId value.

6.16.3.61 void activemq::core::ActiveMQConnection::setDispatchAsync (bool *value*)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

Parameters

value	The value of the dispatch asynchronously option sent to the broker.

6.16.3.62 virtual void activemq::core::ActiveMQConnection::setExceptionListener (cms::ExceptionListener * *listener*) [virtual]

Sets the registered Exception Listener for this connection.

Parameters

listener	pointer to and **ExceptionListener** (p. 996)

Implements **cms::Connection** (p. 728).

6.16.3.63 void activemq::core::ActiveMQConnection::setMessagePrioritySupported (bool *value*)

Set whether or not this factory should create Connection objects with the Message priority support function enabled.

Parameters

value	Boolean indicating if Message priority should be enabled.

6.16.3.64 void activemq::core::ActiveMQConnection::setPassword (const std::string & *password*)

Sets the password that should be used when creating a new connection.

Parameters

password	string

6.16.3.65 void activemq::core::ActiveMQConnection::setPrefetchPolicy (PrefetchPolicy * *policy*)

Sets the **PrefetchPolicy** (p. 1635) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 1635) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy	The new **PrefetchPolicy** (p. 1635) that the ConnectionFactory should clone for Connections.

6.16.3.66 void activemq::core::ActiveMQConnection::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.

Parameters

windowSize	- The size in bytes of the Producers memory window.

6.16.3.67 void activemq::core::ActiveMQConnection::setRedeliveryPolicy (RedeliveryPolicy * *policy*)

Sets the **RedeliveryPolicy** (p. 1744) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 1744) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy	The new **RedeliveryPolicy** (p. 1744) that the ConnectionFactory should clone for Connections.

6.16.3.68 void activemq::core::ActiveMQConnection::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.

Parameters

timeout	- The time to wait for a response.

6.16.3.69 void activemq::core::ActiveMQConnection::setTransportInterruptionProcessingComplete ( )

Indicates that a Connection resource that is processing the transportInterrupted event has completed.

6.16.3.70 void activemq::core::ActiveMQConnection::setUseAsyncSend (bool *value*)

Sets the useAsyncSend option.

Parameters

value	- true to activate, false to disable.

6.16.3.71 void activemq::core::ActiveMQConnection::setUseCompression (bool *value*)

Sets whether Message body compression is enabled.

Parameters

value	Boolean indicating if Message body compression is enabled.

6.16.3.72 void activemq::core::ActiveMQConnection::setUsername (const std::string & *username*)

Sets the username that should be used when creating a new connection.

Parameters

username	string

6.16.3.73 void activemq::core::ActiveMQConnection::signalInterruptionProcessingComplete ( ) [protected]

6.16.3.74 virtual void activemq::core::ActiveMQConnection::start ( ) [virtual]

Starts the service.

Exceptions

CMSException *(p. 640)*	if an internal error occurs while starting.

Implements **cms::Startable** (p. 1964).

6.16.3.75 virtual void activemq::core::ActiveMQConnection::stop ( ) [virtual]

Stops this service.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs while stopping the Service.

Implements **cms::Stoppable** (p. 2017).

6.16.3.76 Pointer<commands::Response> activemq::core::ActiveMQConnection::syncRequest (Pointer< commands::Command > *command,* unsigned int *timeout* = 0)

Sends a synchronous request and returns the response from the broker.

This method converts any error responses it receives into an exception.

Parameters

command	The Command object that is to be sent to the broker.
timeout	The time in milliseconds to wait for a response, default is zero or infinite.

Returns

a Pointer instance to the Response object sent from the Broker.

Exceptions

BrokerException	if the response from the broker is of type ExceptionResponse.
ActiveMQException	if any other error occurs while sending the Command.

6.16.3.77 virtual void activemq::core::ActiveMQConnection::transportInterrupted ( ) [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2178).

6.16.3.78 virtual void activemq::core::ActiveMQConnection::transportResumed ( ) [virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2178).

6.16.3.79 void activemq::core::ActiveMQConnection::waitForTransportInterruptionProcessingToComplete ( ) [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnection.h**

6.17 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 &uri, **const** std::string &username="", **const** std::string &password="")

  Constructor.
- **ActiveMQConnectionFactory** (**const decaf::net::URI** &uri, **const** std::string &username="", **const** std::string &password="")

  Constructor.
- virtual ~**ActiveMQConnectionFactory** () throw ()
- virtual **cms::Connection** * **createConnection** ()

  Creates a connection with the default user identity.
- virtual **cms::Connection** * **createConnection** (**const** std::string &username, **const** std::string &password)

  Creates a connection with the specified user identity.
- virtual **cms::Connection** * **createConnection** (**const** std::string &username, **const** std::string &password, **const** std::string &clientId)

  Creates a connection with the specified user identity.
- void **setUsername** (**const** std::string &username)

  Sets the username that should be used when creating a new connection.
- **const** std::string & **getUsername** () **const**

  Gets the username that this factory will use when creating a new connection instance.
- void **setPassword** (**const** std::string &password)

  Sets the password that should be used when creating a new connection.
- **const** std::string & **getPassword** () **const**

  Gets the password that this factory will use when creating a new connection instance.
- std::string **getClientId** () **const**

Gets the Configured Client Id.

- void **setClientId** (**const** std::string &clientId)

  Sets the Client Id.

- void **setBrokerURI** (**const** std::string &uri)

  Sets the Broker URI that should be used when creating a new connection instance.

- void **setBrokerURI** (**const decaf::net::URI** &uri)

  Sets the Broker URI that should be used when creating a new connection instance.

- **const decaf::net::URI** & **getBrokerURI** () **const**

  Gets the Broker URI that this factory will use when creating a new connection instance.

- void **setExceptionListener** (**cms::ExceptionListener** ∗listener)

  Set an CMS ExceptionListener that will be set on eat connection once it has been created.

- **cms::ExceptionListener** ∗ **getExceptionListener** () **const**

  Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

- void **setPrefetchPolicy** (**PrefetchPolicy** ∗policy)

  *Sets the **PrefetchPolicy** (p. 1635) instance that this factory should use when it creates new Connection instances.*

- **PrefetchPolicy** ∗ **getPrefetchPolicy** () **const**

  *Gets the pointer to the current **PrefetchPolicy** (p. 1635) that is in use by this ConnectionFactory.*

- void **setRedeliveryPolicy** (**RedeliveryPolicy** ∗policy)

  *Sets the **RedeliveryPolicy** (p. 1744) instance that this factory should use when it creates new Connection instances.*

- **RedeliveryPolicy** ∗ **getRedeliveryPolicy** () **const**

  *Gets the pointer to the current **RedeliveryPolicy** (p. 1744) that is in use by this ConnectionFactory.*

- bool **isDispatchAsync** () **const**
- void **setDispatchAsync** (bool value)

  Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

- 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.

- bool **isUseCompression** () **const**

  Gets if the Connection is configured for Message body compression.

- void **setUseCompression** (bool value)

  Sets whether Message body compression is enabled.

- void **setCompressionLevel** (int value)

  Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.

- int **getCompressionLevel** () **const**

  Gets the currently configured Compression level for Message bodies.

- 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.

- bool **isMessagePrioritySupported** () **const**
- void **setMessagePrioritySupported** (bool value)

  Set whether or not this factory should create Connection objects with the Message priority support function enabled.

Static Public Member Functions

- static **cms::Connection** * **createConnection** (**const** std::string &uri, **const** std::string &username, **const** std::string &password, **const** std::string &clientId="")

  Creates a connection with the specified user identity.

Static Public Attributes

- static **const** std::string **DEFAULT_URI**

Protected Member Functions

- virtual **ActiveMQConnection** * **createActiveMQConnection** (**const Pointer**< **transport::Transport** > &transport, **const Pointer**< **decaf::util::Properties** > &properties)

  *Create a new **ActiveMQConnection** (p. 145) instnace using the provided Transport and Properties.*

6.17.1 Constructor & Destructor Documentation

6.17.1.1 activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory ( )

6.17.1.2 activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory (const std::string & *uri,* const std::string & *username* = "", const std::string & *password* = "")

Constructor.

Parameters

url	the URI of the Broker we are connecting to.
username	to authenticate with, defaults to ""
password	to authenticate with, defaults to ""

6.17.1.3 activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory (const decaf::net::URI & *uri,* const std::string & *username* = "", const std::string & *password* = "")

Constructor.

Parameters

uri	the URI of the Broker we are connecting to.
username	to authenticate with, defaults to ""
password	to authenticate with, defaults to ""

6.17.1.4 virtual activemq::core::ActiveMQConnectionFactory::~ActiveMQConnectionFactory ( ) throw () [virtual]

6.17.2 Member Function Documentation

6.17.2.1 virtual ActiveMQConnection* activemq::core::ActiveMQConnectionFactory::createActiveMQConnection (const Pointer< transport::Transport > & *transport*, const Pointer< decaf::util::Properties > & *properties*) [protected, virtual]

Create a new **ActiveMQConnection** (p. 145) instnace using the provided Transport and Properties.

Subclasses can override this to control the actual type of **ActiveMQConnection** (p. 145) that is created.

Parameters

transport	The Transport that the Connection should use to communicate with the Broker.
properties	The Properties that are assigned to the new Connection instance.

Returns

a new **ActiveMQConnection** (p. 145) pointer instance.

Reimplemented in **activemq::core::ActiveMQXAConnectionFactory** (p. 337).

6.17.2.2 virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ( ) [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

CMSException	

Implements **cms::ConnectionFactory** (p. 743).

6.17.2.3 virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & *username*, const std::string & *password*) [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

CMSException	

Implements **cms::ConnectionFactory** (p. 743).

6.17.2.4 virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & *username,* const std::string & *password,* const std::string & *clientId*) [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. 743).

6.17.2.5 static cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & *uri,* const std::string & *username,* const std::string & *password,* const std::string & *clientId* = " ") [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

uri	The URI of the Broker we are connecting to.
username	The name of the user to authenticate with.
password	The password for the user to authenticate with.
clientId	The unique client id to assign to connection, defaults to "".

Exceptions

CMSException.	

6.17.2.6 const decaf::net::URI& activemq::core::ActiveMQConnectionFactory::getBrokerURI ( ) const

Gets the Broker URI that this factory will use when creating a new connection instance.

Returns

brokerURI string

6.17.2.7 std::string activemq::core::ActiveMQConnectionFactory::getClientId ( ) const

Gets the Configured Client Id.

Returns

the clientId.

6.17.2.8 unsigned int activemq::core::ActiveMQConnectionFactory::getCloseTimeout ( ) const

Gets the assigned close timeout for this Connector.

Returns

the close timeout configured in the connection uri

6.17.2.9 int activemq::core::ActiveMQConnectionFactory::getCompressionLevel ( ) const

Gets the currently configured Compression level for Message bodies.

Returns

the int value of the current compression level.

6.17.2.10 cms::ExceptionListener* activemq::core::ActiveMQConnectionFactory::getExceptionListener ( ) const

Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

Returns

a pointer to a CMS ExceptionListener instance or NULL if not set.

6.17.2.11 const std::string& activemq::core::ActiveMQConnectionFactory::getPassword ( ) const

Gets the password that this factory will use when creating a new connection instance.

Returns

password string, "" for default credentials

6.17.2.12 PrefetchPolicy* activemq::core::ActiveMQConnectionFactory::getPrefetchPolicy ( ) const

Gets the pointer to the current **PrefetchPolicy** (p. 1635) that is in use by this ConnectionFactory.

Returns

a pointer to this objects **PrefetchPolicy** (p. 1635).

6.17.2.13 unsigned int activemq::core::ActiveMQConnectionFactory::getProducerWindowSize ( ) const

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.

6.17.2.14 RedeliveryPolicy* activemq::core::ActiveMQConnectionFactory::getRedeliveryPolicy ( ) const

Gets the pointer to the current **RedeliveryPolicy** (p. 1744) that is in use by this ConnectionFactory.

Returns

a pointer to this objects **RedeliveryPolicy** (p. 1744).

6.17.2.15 unsigned int activemq::core::ActiveMQConnectionFactory::getSendTimeout ( ) const

Gets the assigned send timeout for this Connector.

Returns

the send timeout configured in the connection uri

6.17.2.16 const std::string& activemq::core::ActiveMQConnectionFactory::getUsername ( ) const

Gets the username that this factory will use when creating a new connection instance.

Returns

username string, "" for default credentials

6.17.2.17 bool activemq::core::ActiveMQConnectionFactory::isAlwaysSyncSend ( ) const

Gets if the Connection should always send things Synchronously.

Returns

true if sends should always be Synchronous.

6.17.2.18 bool activemq::core::ActiveMQConnectionFactory::isDispatchAsync ( ) const

Returns

The value of the dispatch asynchronously option sent to the broker.

6.17.2.19 bool activemq::core::ActiveMQConnectionFactory::isMessagePrioritySupported ( ) const

Returns

true if the Connections that this factory creates should support the message based priority settings.

6.17.2.20 bool activemq::core::ActiveMQConnectionFactory::isUseAsyncSend ( ) const

Gets if the useAsyncSend option is set.

Returns

true if on false if not.

6.17.2.21 bool activemq::core::ActiveMQConnectionFactory::isUseCompression ( ) const

Gets if the Connection is configured for Message body compression.

Returns

if the Message body will be Compressed or not.

6.17.2.22 void activemq::core::ActiveMQConnectionFactory::setAlwaysSyncSend (bool *value*)

Sets if the Connection should always send things Synchronously.

Parameters

value	true if sends should always be Synchronous.

6.17.2.23 void activemq::core::ActiveMQConnectionFactory::setBrokerURI (const std::string & *uri*)

Sets the Broker URI that should be used when creating a new connection instance.

Parameters

brokerURI	The string form of the Broker URI, this will be converted to a URI object.

6.17.2.24 void activemq::core::ActiveMQConnectionFactory::setBrokerURI (const decaf::net::URI & *uri*)

Sets the Broker URI that should be used when creating a new connection instance.

Parameters

brokerURI	The URI of the broker that this client will connect to.

6.17.2.25 void activemq::core::ActiveMQConnectionFactory::setClientId (const std::string & *clientId*)

Sets the Client Id.

Parameters

clientId	- The new clientId value.

6.17.2.26 void activemq::core::ActiveMQConnectionFactory::setCloseTimeout (unsigned int *timeout*)

Sets the close timeout to use when sending the disconnect request.

Parameters

timeout	- The time to wait for a close message.

6.17.2.27 void activemq::core::ActiveMQConnectionFactory::setCompressionLevel (int *value*)

Sets the Compression level used when Message body compression is enabled, a value of -1 causes the Compression Library to use the default setting which is a balance of speed and compression.

The range of compression levels is [0..9] where 0 indicates best speed and 9 indicates best compression.

Parameters

value	A signed int value that controls the compression level.

6.17.2.28 void activemq::core::ActiveMQConnectionFactory::setDispatchAsync (bool *value*)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

Parameters

value	The value of the dispatch asynchronously option sent to the broker.

6.17.2.29 void activemq::core::ActiveMQConnectionFactory::setExceptionListener (cms::ExceptionListener * *listener*)

Set an CMS ExceptionListener that will be set on eat connection once it has been created.

The factory does not take ownership of this pointer, the client must ensure that its lifetime is scoped to the connection that it is applied to.

Parameters

listener	The listener to set on the connection or NULL for no listener.

6.17.2.30 void activemq::core::ActiveMQConnectionFactory::setMessagePrioritySupported (bool *value*)

Set whether or not this factory should create Connection objects with the Message priority support function enabled.

Parameters

value	Boolean indicating if Message priority should be enabled.

6.17.2.31 void activemq::core::ActiveMQConnectionFactory::setPassword (const std::string & *password*)

Sets the password that should be used when creating a new connection.

Parameters

password	string

6.17.2.32 void activemq::core::ActiveMQConnectionFactory::setPrefetchPolicy (PrefetchPolicy * *policy*)

Sets the **PrefetchPolicy** (p. 1635) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 1635) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy	The new **PrefetchPolicy** (p. 1635) that the ConnectionFactory should clone for Connections.

6.17.2.33 void activemq::core::ActiveMQConnectionFactory::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.

Parameters

windowSize	- The size in bytes of the Producers memory window.

6.17.2.34 void activemq::core::ActiveMQConnectionFactory::setRedeliveryPolicy (RedeliveryPolicy * *policy*)

Sets the **RedeliveryPolicy** (p. 1744) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 1744) passed becomes the property of the factory and will be deleted when the factory is destroyed.

Parameters

policy	The new **RedeliveryPolicy** (p. 1744) that the ConnectionFactory should clone for Connections.

6.17.2.35 void activemq::core::ActiveMQConnectionFactory::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.

Parameters

timeout	- The time to wait for a response.

6.17.2.36 void activemq::core::ActiveMQConnectionFactory::setUseAsyncSend (bool *value*)

Sets the useAsyncSend option.

Parameters

value	- true to activate, false to disable.

6.17.2.37 void activemq::core::ActiveMQConnectionFactory::setUseCompression (bool *value*)

Sets whether Message body compression is enabled.

Parameters

value	Boolean indicating if Message body compression is enabled.

6.17.2.38 void activemq::core::ActiveMQConnectionFactory::setUsername (const std::string & *username*)

Sets the username that should be used when creating a new connection.

Parameters

username	string

6.17.3 Field Documentation

6.17.3.1 const std::string activemq::core::ActiveMQConnectionFactory::DEFAULT_URI [static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionFactory.h**

6.18 activemq::core::ActiveMQConnectionMetaData Class Reference

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 145) class.

`#include <src/main/activemq/core/ActiveMQConnectionMetaData.h>`

Inheritance diagram for activemq::core::ActiveMQConnectionMetaData:

Public Member Functions

- **ActiveMQConnectionMetaData** ()
- virtual ~**ActiveMQConnectionMetaData** () throw ()
- virtual std::string **getCMSVersion** () **const**

  Gets the CMS API version.
- virtual int **getCMSMajorVersion** () **const**

  Gets the CMS major version number.
- virtual int **getCMSMinorVersion** () **const**

  Gets the CMS minor version number.

- virtual std::string **getCMSProviderName** () **const**

  Gets the CMS provider name.
- virtual std::string **getProviderVersion** () **const**

  Gets the CMS provider version.
- virtual int **getProviderMajorVersion** () **const**

  Gets the CMS provider major version number.
- virtual int **getProviderMinorVersion** () **const**

  Gets the CMS provider minor version number.
- virtual std::vector< std::string > **getCMSXPropertyNames** () **const**

  Gets an Vector of the CMSX property names.

6.18.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 145) class.

Since

3.0

6.18.2 Constructor & Destructor Documentation

6.18.2.1 activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData ( )

6.18.2.2 virtual activemq::core::ActiveMQConnectionMetaData::~ActiveMQConnectionMetaData ( ) throw () [virtual]

6.18.3 Member Function Documentation

6.18.3.1 virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion ( ) const [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. 760).

6.18.3.2 virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion ( ) const [virtual]

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

CMSException	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 760).

6.18.3.3 virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName ( ) const [virtual]

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

CMSException	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 760).

6.18.3.4 virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion ( ) const [virtual]

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

CMSException	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 761).

6.18.3.5 virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames ( ) const [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. 761).

6.18.3.6 virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion ( ) const [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. 761).

6.18.3.7 virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion ( ) const [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. 761).

6.18.3.8 virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion ( ) const [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. 762).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionMetaData.h**

6.19 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 an 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_PREFECTCHSIZE, **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_ALWAYSSYNCSEND**,
  CONNECTION_USEASYNCSEND, **CONNECTION_USECOMPRESSION**, **CONNECTION_DISPATCHASYNC**, **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)

6.19.1 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.

6.19.2 Member Enumeration Documentation

6.19.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

ACK_TYPE_DELIVERED

ACK_TYPE_POISON

ACK_TYPE_CONSUMED

ACK_TYPE_REDELIVERED

ACK_TYPE_INDIVIDUAL

6.19.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

DESTINATION_ADD_OPERATION

DESTINATION_REMOVE_OPERATION

6.19.2.3 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

6.19.2.4 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

6.19.2.5 enum activemq::core::ActiveMQConstants::URIParam

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_ALWAYSSYNCSEND

CONNECTION_USEASYNCSEND

CONNECTION_USECOMPRESSION

CONNECTION_DISPATCHASYNC

PARAM_USERNAME

PARAM_PASSWORD

PARAM_CLIENTID

NUM_PARAMS

6.19.3 Member Function Documentation

6.19.3.1 static DestinationOption activemq::core::ActiveMQConstants::toDestinationOption (const std::string & *option*) [inline, static]

6.19.3.2 static const std::string& activemq::core::ActiveMQConstants::toString (const DestinationOption *option*) [inline, static]

6.19.3.3 static const std::string& activemq::core::ActiveMQConstants::toString (const URIParam *option*) [inline, static]

6.19.3.4 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**

6.20 activemq::core::ActiveMQConsumer Class Reference

`#include <src/main/activemq/core/ActiveMQConsumer.h>`

Inheritance diagram for activemq::core::ActiveMQConsumer:

Public Member Functions

- **ActiveMQConsumer** (**ActiveMQSession** *session, **const Pointer**< **commands::ConsumerId** > &id, **const Pointer**< **commands::ActiveMQDestination** > &destination, **const** std::string &name, **const** std::string &selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, **cms::MessageListener** *listener)

  Constructor.
- virtual ~**ActiveMQConsumer** () throw ()
- virtual void **start** ()

  Starts the service.
- virtual void **stop** ()

  Stops this service.
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- virtual **cms::Message** * **receive** ()

  Synchronously Receive a Message.
- virtual **cms::Message** * **receive** (int millisecs)

  Synchronously Receive a Message, time out after defined interval.
- virtual **cms::Message** * **receiveNoWait** ()

  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)

  Sets the MessageListener that this class will send notifs on.
- virtual **cms::MessageListener** ∗ **getMessageListener** () **const**

  Gets the MessageListener that this class will send mew Message notification events to.
- virtual std::string **getMessageSelector** () **const**

  Gets this message consumer's message selector expression.
- virtual void **acknowledge** (**const Pointer**< **commands::MessageDispatch** > &**dispatch**)
- virtual void **dispatch** (**const Pointer**< **MessageDispatch** > &message)

  Dispatches a message to a particular consumer.
- void **acknowledge** ()

  Method called to acknowledge all messages that have been received so far.
- void **commit** ()

  Called to Commit the current set of messages in this Transaction.
- void **rollback** ()

  Called to Roll back the current set of messages in this Transaction.
- void **doClose** ()

  Performs the actual close operation on this consumer.
- void **dispose** ()

  Cleans up this objects internal resources.
- **const Pointer**
  < **commands::ConsumerInfo** > & **getConsumerInfo** () **const**

  Get the Consumer information for this consumer.
- **const Pointer**
  < **commands::ConsumerId** > & **getConsumerId** () **const**

  Get the Consumer Id for this consumer.
- bool **isClosed** () **const**
- bool **isSynchronizationRegistered** () **const**

  *Has this Consumer Transaction **Synchronization** (p. 2056) been added to the transaction.*
- void **setSynchronizationRegistered** (bool value)

  *Sets the **Synchronization** (p. 2056) 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** ()

  Forces this consumer to send all pending acks to the broker.
- void **clearMessagesInProgress** ()

  Called on a Failover to clear any pending messages.
- void **inProgressClearRequired** ()

  Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.
- 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.
- int **getMessageAvailableCount** () **const**
- void **setRedeliveryPolicy** (**RedeliveryPolicy** ∗policy)

  *Sets the **RedeliveryPolicy** (p. 1744) this Consumer should use when a rollback is performed on a transacted Consumer.*
- **RedeliveryPolicy** ∗ **getRedeliveryPolicy** () **const**

  Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.
- void **setFailureError** (**decaf::lang::Exception** ∗error)

  Sets the Exception that has caused this Consumer to be in a failed state.
- **decaf::lang::Exception** ∗ **getFailureError** () **const**

  Gets the error that caused this Consumer to be in a Failed state, or NULL if there is no Error.

Protected Member Functions

- **Pointer< MessageDispatch > dequeue** (long long timeout)

  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.

6.20.1 Constructor & Destructor Documentation

6.20.1.1 activemq::core::ActiveMQConsumer::ActiveMQConsumer (ActiveMQSession * *session,* const Pointer< commands::ConsumerId > & *id,* const Pointer< commands::ActiveMQDestination > & *destination,* const std::string & *name,* const std::string & *selector,* int *prefetch,* int *maxPendingMessageCount,* bool *noLocal,* bool *browser,* bool *dispatchAsync,* cms::MessageListener * *listener*)

Constructor.

6.20.1.2 virtual activemq::core::ActiveMQConsumer::~ActiveMQConsumer ( ) throw () [virtual]

6.20.2 Member Function Documentation

6.20.2.1 virtual void activemq::core::ActiveMQConsumer::acknowledge (const Pointer< commands::MessageDispatch > & *dispatch*) [virtual]

6.20.2.2 void activemq::core::ActiveMQConsumer::acknowledge ( )

Method called to acknowledge all messages that have been received so far.

Exceptions

CMSException	if an error occurs while ack'ing the message.

6.20.2.3 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.

6.20.2.4 void activemq::core::ActiveMQConsumer::beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > & *dispatch*) [protected]

Pre-consume processing.

Parameters

dispatch	- the message being consumed.

6.20.2.5 void activemq::core::ActiveMQConsumer::clearMessagesInProgress ( )

Called on a Failover to clear any pending messages.

6.20.2.6 virtual void activemq::core::ActiveMQConsumer::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

CMSException	- If an error occurs while the resource is being closed.

Implements **cms::Closeable** (p. 633).

6.20.2.7 void activemq::core::ActiveMQConsumer::commit ( )

Called to Commit the current set of messages in this Transaction.

Exceptions

ActiveMQException	if an error occurs while performing the operation.

6.20.2.8 void activemq::core::ActiveMQConsumer::deliverAcks ( )

Forces this consumer to send all pending acks to the broker.

Exceptions

ActiveMQException	if an error occurs while performing the operation.

6.20.2.9 Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue (long long *timeout*) [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.

6.20.2.10 virtual void activemq::core::ActiveMQConsumer::dispatch (const Pointer< MessageDispatch > & *message*) [virtual]

Dispatches a message to a particular consumer.

Parameters

message	The message to be dispatched to a waiting consumer.

Implements **activemq::core::Dispatcher** (p. 956).

6.20.2.11 void activemq::core::ActiveMQConsumer::dispose ( )

Cleans up this objects internal resources.

Exceptions

ActiveMQException	if an error occurs while performing the operation.

6.20.2.12 void activemq::core::ActiveMQConsumer::doClose ( )

Performs the actual close operation on this consumer.

Exceptions

ActiveMQException	if an error occurs while performing the operation.

6.20.2.13 const Pointer<commands::ConsumerId>& activemq::core::ActiveMQConsumer::getConsumerId ( ) const

Get the Consumer Id for this consumer.

Returns

Reference to a Consumer Id Object

6.20.2.14 const Pointer<commands::ConsumerInfo>& activemq::core::ActiveMQConsumer::getConsumerInfo ( ) const

Get the Consumer information for this consumer.

Returns

Reference to a Consumer Info Object

6.20.2.15 decaf::lang::Exception* activemq::core::ActiveMQConsumer::getFailureError ( ) const

Gets the error that caused this Consumer to be in a Failed state, or NULL if there is no Error.

Returns

pointer to the error that faulted this Consumer or NULL.

6.20.2.16 long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId ( ) const

Gets the currently set Last Delivered Sequence Id.

Returns

long long containing the sequence id of the last delivered Message.

6.20.2.17 int activemq::core::ActiveMQConsumer::getMessageAvailableCount ( ) const

Returns

the number of Message's this consumer is waiting to Dispatch.

6.20.2.18 virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener ( ) const [virtual]

Gets the MessageListener that this class will send mew Message notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1456).

6.20.2.19 virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector ( ) const [virtual]

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1456).

6.20.2.20 RedeliveryPolicy* activemq::core::ActiveMQConsumer::getRedeliveryPolicy ( ) const

Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

Returns

a Pointer to a **RedeliveryPolicy** (p. 1744) that is in use by this Consumer.

6.20.2.21 void activemq::core::ActiveMQConsumer::inProgressClearRequired ( )

Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.

6.20.2.22 bool activemq::core::ActiveMQConsumer::isClosed ( ) const

Returns

if this Consumer has been closed.

6.20.2.23 bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered ( ) const

Has this Consumer Transaction **Synchronization** (p. 2056) been added to the transaction.

Returns

true if the synchronization has been added.

6.20.2.24 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.

6.20.2.25 virtual cms::Message* activemq::core::ActiveMQConsumer::receive ( ) [virtual]

Synchronously Receive a Message.

Returns

new message which the caller owns and must delete.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1457).

6.20.2.26 virtual cms::Message* activemq::core::ActiveMQConsumer::receive (int *millisecs*) [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

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1457).

6.20.2.27 virtual cms::Message* activemq::core::ActiveMQConsumer::receiveNoWait ( ) [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

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1457).

6.20.2.28 void activemq::core::ActiveMQConsumer::rollback ( )

Called to Roll back the current set of messages in this Transaction.

Exceptions

ActiveMQException	if an error occurs while performing the operation.

6.20.2.29 void activemq::core::ActiveMQConsumer::setFailureError (decaf::lang::Exception * *error*)

Sets the Exception that has caused this Consumer to be in a failed state.

Parameters

error	The error that is to be thrown when a Receive call is made.

6.20.2.30 void activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId (long long *value*)

Sets the value of the Last Delivered Sequence Id.

Parameters

value	The new value to assign to the Last Delivered Sequence Id property.

6.20.2.31 virtual void activemq::core::ActiveMQConsumer::setMessageListener (cms::MessageListener * *listener*) [virtual]

Sets the MessageListener that this class will send notifs on.

Parameters

listener	The listener of messages received by this consumer.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1458).

6.20.2.32 void activemq::core::ActiveMQConsumer::setRedeliveryPolicy (RedeliveryPolicy * *policy*)

Sets the **RedeliveryPolicy** (p. 1744) this Consumer should use when a rollback is performed on a transacted Consumer.

The Consumer takes ownership of the passed pointer. The Consumer's redelivery policy can never be null, a call to this method with a NULL pointer is ignored.

Parameters

policy	Pointer to a Redelivery Policy object that his Consumer will use.

6.20.2.33 void activemq::core::ActiveMQConsumer::setSynchronizationRegistered (bool *value*)

Sets the **Synchronization** (p. 2056) Registered state of this consumer.

Parameters

value	- true if registered false otherwise.

6.20.2.34 virtual void activemq::core::ActiveMQConsumer::start ( ) [virtual]

Starts the service.

Exceptions

CMSException	if an internal error occurs while starting.

Implements **cms::Startable** (p. 1964).

6.20.2.35 virtual void activemq::core::ActiveMQConsumer::stop ( ) [virtual]

Stops this service.

Exceptions

CMSException	- if an internal error occurs while stopping the Service.

Implements **cms::Stoppable** (p. 2017).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConsumer.h**

6.21 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** &)

6.21.1 Constructor & Destructor Documentation

6.21.1.1 activemq::library::ActiveMQCPP::ActiveMQCPP ( ) [inline, protected]

6.21.1.2 activemq::library::ActiveMQCPP::ActiveMQCPP (const ActiveMQCPP &) [protected]

6.21.1.3 virtual activemq::library::ActiveMQCPP::~ActiveMQCPP ( ) [inline, virtual]

6.21.2 Member Function Documentation

6.21.2.1 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.

6.21.2.2 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.

6.21.2.3 ActiveMQCPP& activemq::library::ActiveMQCPP::operator= (const ActiveMQCPP &) [protected]

6.21.2.4 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**

6.22 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** () throw ()
- virtual **ActiveMQDestination** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual std::string **toString** () **const**

  *Returns a string containing the information for this **DataStructure** (p. 877) 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_QUALIFED_PREFIX**
- static **const** std::string **TEMP_TOPIC_QUALIFED_PREFIX**

6.22.1 Constructor & Destructor Documentation

6.22.1.1 activemq::commands::ActiveMQDestination::ActiveMQDestination ( )

6.22.1.2 activemq::commands::ActiveMQDestination::ActiveMQDestination (const std::string & *physicalName*)

6.22.1.3 virtual activemq::commands::ActiveMQDestination::~ActiveMQDestination ( ) throw () [virtual]

6.22.2 Member Function Documentation

6.22.2.1 virtual ActiveMQDestination* activemq::commands::ActiveMQDestination::cloneDataStructure ( ) const [inline, virtual]

Clone this obbject 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. 878).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 294), **activemq::commands::ActiveMQQueue** (p. 250), **activemq::commands::ActiveMQTopic** (p. 322), **activemq::commands::ActiveMQTempQueue** (p. 301), and **activemq::commands::ActiveMQTempTopic** (p. 308).

6.22.2.2 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. 879).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 295), **activemq::commands::ActiveMQQueue** (p. 250), **activemq::commands::ActiveMQTopic** (p. 322), **activemq::commands::ActiveMQTempQueue** (p. 301), and **activemq::commands::ActiveMQTempTopic** (p. 308).

Referenced by activemq::commands::ActiveMQTempDestination::copyDataStructure().

6.22.2.3 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. 191) instance.

6.22.2.4 static std::string activemq::commands::ActiveMQDestination::createTemporaryName (const std::string & *clientId*) [inline, static]

Create a temporary name from the clientId.

Parameters

clientId	

Returns

6.22.2.5 virtual bool activemq::commands::ActiveMQDestination::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 295), **activemq::commands::ActiveMQQueue** (p. 250), **activemq::commands::ActiveMQTopic** (p. 323), **activemq::commands::ActiveMQTempQueue** (p. 301), and **activemq::commands::ActiveMQTempTopic** (p. 309).

Referenced by activemq::commands::ActiveMQTempDestination::equals().

6.22.2.6 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

6.22.2.7 virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination ( ) const [inline, virtual]

Returns

the **cms::Destination** (p. 936) interface pointer that the objects that derive from this class implement.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 251), **activemq::commands::ActiveMQTopic** (p. 323), **activemq::commands::ActiveMQTempQueue** (p. 302), and **activemq::commands::ActiveMQTempTopic** (p. 309).

6.22.2.8 virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 295), **activemq::commands::ActiveMQQueue** (p. 251), **activemq::commands::ActiveMQTopic** (p. 323), **activemq::commands::ActiveMQTempQueue** (p. 302), and **activemq::commands::ActiveMQTempTopic** (p. 309).

6.22.2.9 virtual cms::Destination::DestinationType activemq::commands::ActiveMQDestination::getDestinationType ( ) const [pure virtual]

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implemented in **activemq::commands::ActiveMQQueue** (p. 251), **activemq::commands::ActiveMQTopic** (p. 324), **activemq::commands::ActiveMQTempQueue** (p. 303), and **activemq::commands::ActiveMQTempTopic** (p. 310).

6.22.2.10 const activemq::util::ActiveMQProperties& activemq::commands::ActiveMQDestination::getOptions ( ) const [inline]

Returns

a reference (const) to the options properties for this Destination.

6.22.2.11 virtual std::string activemq::commands::ActiveMQDestination::getOrderedTarget ( ) const [inline, virtual]

Returns

Returns the orderedTarget.

6.22.2.12 virtual const std::string& activemq::commands::ActiveMQDestination::getPhysicalName ( ) const [inline, virtual]

Fetch this destination's physical name.

Returns

const string containing the name

6.22.2.13 virtual std::string& activemq::commands::ActiveMQDestination::getPhysicalName ( ) [inline, virtual]

6.22.2.14 virtual bool activemq::commands::ActiveMQDestination::isAdvisory ( ) const [inline, virtual]

Returns

Returns the advisory.

6.22.2.15 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.

6.22.2.16 virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory ( ) const [inline, virtual]

Returns

true if this is a destination for Connection advisories

6.22.2.17 virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory ( ) const [inline, virtual]

Returns

true if this is a destination for Consumer advisories

6.22.2.18 virtual bool activemq::commands::ActiveMQDestination::isExclusive ( ) const [inline, virtual]

Returns

Returns the exclusive.

6.22.2.19 virtual bool activemq::commands::ActiveMQDestination::isOrdered ( ) const [inline, virtual]

Returns

Returns the ordered.

6.22.2.20 virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory ( ) const [inline, virtual]

Returns

true if this is a destination for Producer advisories

6.22.2.21 virtual bool activemq::commands::ActiveMQDestination::isQueue ( ) const [inline, virtual]

Returns true if a Queue Destination.

Returns

true/false

6.22.2.22 virtual bool activemq::commands::ActiveMQDestination::isTemporary ( ) const [inline, virtual]

Returns true if a temporary Destination.

Returns

true/false

References cms::Destination::TEMPORARY_QUEUE, and cms::Destination::TEMPORARY_TOPIC.

6.22.2.23 virtual bool activemq::commands::ActiveMQDestination::isTopic ( ) const [inline, virtual]

Returns true if a Topic Destination.

Returns

true/false

References cms::Destination::TEMPORARY_TOPIC, and cms::Destination::TOPIC.

6.22.2.24 virtual bool activemq::commands::ActiveMQDestination::isWildcard ( ) const [inline, virtual]

Returns

true if the destination matches multiple possible destinations

6.22.2.25 virtual void activemq::commands::ActiveMQDestination::setAdvisory (bool *advisory*) [inline, virtual]

Parameters

advisory	The advisory to set.

6.22.2.26 virtual void activemq::commands::ActiveMQDestination::setExclusive (bool *exclusive*) [inline, virtual]

Parameters

exclusive	The exclusive to set.

6.22.2.27 virtual void activemq::commands::ActiveMQDestination::setOrdered (bool *ordered*) [inline, virtual]

Parameters

ordered	The ordered to set.

6.22.2.28 virtual void activemq::commands::ActiveMQDestination::setOrderedTarget (const std::string & *orderedTarget*) [inline, virtual]

Parameters

orderedTarget	The orderedTarget to set.

6.22.2.29 virtual void activemq::commands::ActiveMQDestination::setPhysicalName (const std::string & *physicalName*) [virtual]

Set this destination's physical name.

Returns

const string containing the name

6.22.2.30 virtual std::string activemq::commands::ActiveMQDestination::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

Reimplemented in **activemq::commands::ActiveMQTempDestination** (p. 296), **activemq::commands::ActiveMQQueue** (p. 252), **activemq::commands::ActiveMQTopic** (p. 324), **activemq::commands::ActiveMQTempQueue** (p. 303), and **activemq::commands::ActiveMQTempTopic** (p. 310).

6.22.3 Field Documentation

6.22.3.1 bool activemq::commands::ActiveMQDestination::advisory [protected]

6.22.3.2 const std::string activemq::commands::ActiveMQDestination::ADVISORY_PREFIX [static, protected]

prefix for Advisory message destinations

6.22.3.3 const std::string activemq::commands::ActiveMQDestination::COMPOSITE_SEPARATOR [static, protected]

6.22.3.4 const std::string activemq::commands::ActiveMQDestination::CONNECTION_ADVISORY_PREFIX [static, protected]

prefix for connection advisory destinations

6.22.3.5 const std::string activemq::commands::ActiveMQDestination::CONSUMER_ADVISORY_PREFIX [static, protected]

prefix for consumer advisory destinations

6.22.3.6 const std::string activemq::commands::ActiveMQDestination::DEFAULT_ORDERED_TARGET [static, protected]

The default target for ordered destinations.

6.22.3.7 bool activemq::commands::ActiveMQDestination::exclusive [protected]

6.22.3.8 const unsigned char activemq::commands::ActiveMQDestination::ID_ACTIVEMQDESTINATION = 0 [static]

6.22.3.9 util::ActiveMQProperties activemq::commands::ActiveMQDestination::options [protected]

6.22.3.10 bool activemq::commands::ActiveMQDestination::ordered [protected]

6.22.3.11 std::string activemq::commands::ActiveMQDestination::orderedTarget [protected]

6.22.3.12 std::string activemq::commands::ActiveMQDestination::physicalName [protected]

6.22.3.13 const std::string activemq::commands::ActiveMQDestination::PRODUCER_ADVISORY_PREFIX [static, protected]

prefix for producer advisory destinations

6.22.3.14 const std::string activemq::commands::ActiveMQDestination::QUEUE_QUALIFIED_PREFIX [static, protected]

6.22.3.15 const std::string activemq::commands::ActiveMQDestination::TEMP_POSTFIX [static, protected]

6.22.3.16 const std::string activemq::commands::ActiveMQDestination::TEMP_PREFIX [static, protected]

6.22.3.17 const std::string activemq::commands::ActiveMQDestination::TEMP_QUEUE_QUALIFED_PREFIX [static, protected]

6.22.3.18 const std::string activemq::commands::ActiveMQDestination::TEMP_TOPIC_QUALIFED_PREFIX [static, protected]

6.22.3.19 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**

6.23 activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 200).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQDestinationMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ∼**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.23.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 200).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.23.2 Constructor & Destructor Documentation

6.23.2.1 activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ( ) [inline]

6.23.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::∼ActiveMQDestinationMarshaller ( ) [inline, virtual]

6.23.3 Member Function Documentation

6.23.3.1 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 256), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 326), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 297).

6.23.3.2 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 256), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 326), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298).

6.23.3.3 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 257), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 326), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298).

6.23.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 257), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 306), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 313), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 327), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299).

6.23.3.5 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 258), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 306), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 313), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 327), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQDestinationMarshaller.h**

6.24 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::CMSException convertToCMSException** () **const**

  Converts this exception to a new CMSException.

6.24.1 Constructor & Destructor Documentation

6.24.1.1 activemq::exceptions::ActiveMQException::ActiveMQException ( ) throw ()

Default Constructor.

6.24.1.2 activemq::exceptions::ActiveMQException::ActiveMQException (const ActiveMQException & *ex*) throw ()

Copy Constructor.

Parameters

ex	The Exception whose internal data is copied into this instance.

6.24.1.3 activemq::exceptions::ActiveMQException::ActiveMQException (const decaf::lang::Exception & *ex*) throw ()

Copy Constructor.

Parameters

ex	The Exception whose internal data is copied into this instance.

6.24.1.4 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.

6.24.1.5 virtual activemq::exceptions::ActiveMQException::~ActiveMQException ( ) throw () [virtual]

6.24.2 Member Function Documentation

6.24.2.1 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. 992).

Reimplemented in **activemq::exceptions::BrokerException** (p. 437), and **activemq::exceptions::ConnectionFailedException** (p. 745).

6.24.2.2 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**

6.25 activemq::commands::ActiveMQMapMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQMapMessage.h>`

Inheritance diagram for activemq::commands::ActiveMQMapMessage:

Public Member Functions

- **ActiveMQMapMessage** ()
- virtual ∼**ActiveMQMapMessage** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual bool **isMarshalAware** () **const**

  Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual **ActiveMQMapMessage** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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)

  Called before marshaling is started to prepare the object to be marshaled.
- virtual std::string **toString** () **const**

  *Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) 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 **cms::MapMessage** ∗ **clone** () **const**

  Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual bool **isEmpty** () **const**

  *Returns true if there are no values stored in the **MapMessage** (p. 1379) body.*

  Returns

  *true if the body of the **MapMessage** (p. 1379) contains no elements.*

  Exceptions

CMSException (p. 640)	*if the operation fails due to an internal error.*

- virtual std::vector< std::string > **getMapNames** () **const**

  *Returns an Enumeration of all the names in the **MapMessage** (p. 1379) object.*

  Returns

  *STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 1379)*

  Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*

- virtual bool **itemExists** (**const** std::string &name) **const**

  *Indicates whether an item exists in this **MapMessage** (p. 1379) object.*

  Parameters

name	*String name of the Object in question*

  Returns

  boolean value indicating if the name is in the map

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*

- virtual bool **getBoolean** (**const** std::string &name) **const**

  Returns the Boolean value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setBoolean** (**const** std::string &name, bool value)

  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

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageNotWritableException	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual unsigned char **getByte** (**const** std::string &name) **const**

  Returns the Byte value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setByte** (**const** std::string &name, unsigned char value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual std::vector< unsigned char > **getBytes** (**const** std::string &name) **const**

  Returns the Bytes value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setBytes** (**const** std::string &name, **const** std::vector< unsigned char > &value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual char **getChar** (**const** std::string &name) **const**

  Returns the Char value of the Specified name.

Parameters

name	*name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setChar** (**const** std::string &name, char value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual double **getDouble** (**const** std::string &name) **const**

  Returns the Double value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setDouble** (**const** std::string &name, double value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual float **getFloat** (**const** std::string &name) **const**

  Returns the Float value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setFloat** (**const** std::string &name, float value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual int **getInt** (**const** std::string &name) **const**

  Returns the Int value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setInt** (**const** std::string &name, int value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual long long **getLong** (**const** std::string &name) **const**

  Returns the Long value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setLong** (**const** std::string &name, long long value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual short **getShort** (**const** std::string &name) **const**

  Returns the Short value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setShort** (**const** std::string &name, short value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

- virtual std::string **getString** (**const** std::string &name) **const**

  Returns the String value of the Specified name.

Parameters

name	*Name of the value to fetch from the map*

Exceptions

CMSException (p. 640)	*- if the operation fails due to an internal error.*
MessageFormatException (p. 1478)	*- if this type conversion is invalid.*

- virtual void **setString** (**const** std::string &name, **const** std::string &value)

  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. 640)	*- if the operation fails due to an internal error.*
MessageNotWriteable-Exception (p. 1490)	*- if the **Message** (p. 1426) is in Read-only Mode.*

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQMAPMESSAGE** = 25

Protected Member Functions

- **util::PrimitiveMap** & **getMap** ()

  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**
- virtual void **checkMapIsUnmarshalled** () **const**

  Performs the unmarshal on the Map if needed, otherwise just returns.

6.25.1 Constructor & Destructor Documentation

6.25.1.1 activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ( )

6.25.1.2 virtual activemq::commands::ActiveMQMapMessage::~ActiveMQMapMessage ( ) throw () [virtual]

6.25.2 Member Function Documentation

6.25.2.1 virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal (wireformat::WireFormat * *wireFormat*) [virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

wireFormat	The wireformat object to control marshaling

Exceptions

IOException	if an I/O error occurs.

Implements **activemq::wireformat::MarshalAware** (p. 1390).

6.25.2.2 virtual void activemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled ( ) const [protected, virtual]

Performs the unmarshal on the Map if needed, otherwise just returns.

Exceptions

NullPointerException	if the internal Map is Null.

6.25.2.3 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.

Exceptions

CMSException	- if an internal error occurs.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230).

6.25.2.4 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

Implements **cms::Message** (p. 1430).

6.25.2.5 virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1417).

6.25.2.6 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. 1417).

6.25.2.7 virtual bool activemq::commands::ActiveMQMapMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230).

6.25.2.8 virtual bool activemq::commands::ActiveMQMapMessage::getBoolean (const std::string & *name*) const [virtual]

Returns the Boolean value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1381).

6.25.2.9 virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte (const std::string & *name*) const [virtual]

Returns the Byte value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1382).

6.25.2.10 virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes (const std::string & *name*) const [virtual]

Returns the Bytes value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1382).

6.25.2.11 virtual char activemq::commands::ActiveMQMapMessage::getChar (const std::string & *name*) const [virtual]

Returns the Char value of the Specified name.

Parameters

name	name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1382).

6.25.2.12 virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1419).

6.25.2.13 virtual double activemq::commands::ActiveMQMapMessage::getDouble (const std::string & *name*) const [virtual]

Returns the Double value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1382).

6.25.2.14 virtual float activemq::commands::ActiveMQMapMessage::getFloat (const std::string & *name*) const [virtual]

Returns the Float value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1383).

6.25.2.15 virtual int activemq::commands::ActiveMQMapMessage::getInt (const std::string & *name*) const [virtual]

Returns the Int value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1383).

6.25.2.16 virtual long long activemq::commands::ActiveMQMapMessage::getLong (const std::string & *name*) const [virtual]

Returns the Long value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1383).

6.25.2.17 util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap ( ) [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 PrimtiveMap;

Exceptions

NullPointerException	if the internal Map is Null.

6.25.2.18 const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap ( ) const [protected]

6.25.2.19 virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames ( ) const [virtual]

Returns an Enumeration of all the names in the **MapMessage** (p. 1379) object.

Returns

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 1379)

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.

Implements **cms::MapMessage** (p. 1384).

6.25.2.20 virtual short activemq::commands::ActiveMQMapMessage::getShort (const std::string & *name*) const [virtual]

Returns the Short value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1384).

6.25.2.21 virtual std::string activemq::commands::ActiveMQMapMessage::getString (const std::string & *name*) const [virtual]

Returns the String value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 1384).

6.25.2.22 virtual bool activemq::commands::ActiveMQMapMessage::isEmpty ( ) const [virtual]

Returns true if there are no values stored in the **MapMessage** (p. 1379) body.

Returns

true if the body of the **MapMessage** (p. 1379) contains no elements.

Exceptions

CMSException *(p. 640)*	if the operation fails due to an internal error.

Implements **cms::MapMessage** (p. 1385).

6.25.2.23 virtual bool activemq::commands::ActiveMQMapMessage::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::Message** (p. 1421).

6.25.2.24 virtual bool activemq::commands::ActiveMQMapMessage::itemExists (const std::string & *name*) const [virtual]

Indicates whether an item exists in this **MapMessage** (p. 1379) object.

Parameters

name	String name of the Object in question

Returns

boolean value indicating if the name is in the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.

Implements **cms::MapMessage** (p. 1385).

6.25.2.25 virtual void activemq::commands::ActiveMQMapMessage::setBoolean (const std::string & *name,* bool *value*) [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

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageNotWritable-Exception	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1385).

6.25.2.26 virtual void activemq::commands::ActiveMQMapMessage::setByte (const std::string & *name,* unsigned char *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1385).

6.25.2.27 virtual void activemq::commands::ActiveMQMapMessage::setBytes (const std::string & *name*, const std::vector< unsigned char > & *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1386).

6.25.2.28 virtual void activemq::commands::ActiveMQMapMessage::setChar (const std::string & *name*, char *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1386).

6.25.2.29 virtual void activemq::commands::ActiveMQMapMessage::setDouble (const std::string & *name*, double *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1386).

6.25.2.30 virtual void activemq::commands::ActiveMQMapMessage::setFloat (const std::string & *name,* float *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1387).

6.25.2.31 virtual void activemq::commands::ActiveMQMapMessage::setInt (const std::string & *name,* int *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1387).

6.25.2.32 virtual void activemq::commands::ActiveMQMapMessage::setLong (const std::string & *name,* long long *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1387).

6.25.2.33 virtual void activemq::commands::ActiveMQMapMessage::setShort (const std::string & *name,* short *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1388).

6.25.2.34 virtual void activemq::commands::ActiveMQMapMessage::setString (const std::string & *name,* const std::string & *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implements **cms::MapMessage** (p. 1388).

6.25.2.35 virtual std::string activemq::commands::ActiveMQMapMessage::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1424).

6.25.3 Field Documentation

6.25.3.1 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**

6.26 activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessage-Marshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 220).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ-MapMessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual ∼**ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.26.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 220).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.26.2 Constructor & Destructor Documentation

6.26.2.1 activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]

6.26.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::~ActiveMQMapMessageMarshaller ( ) [inline, virtual]

6.26.3 Member Function Documentation

6.26.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.26.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.26.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.26.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.26.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.26.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.26.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQMapMessageMarshaller.h**

6.27 activemq::commands::ActiveMQMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQMessage.h>`

Inheritance diagram for activemq::commands::ActiveMQMessage:

Public Member Functions

- **ActiveMQMessage** ()
- virtual ~**ActiveMQMessage** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- 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 obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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

6.27.1 Constructor & Destructor Documentation

6.27.1.1 activemq::commands::ActiveMQMessage::ActiveMQMessage ( )

6.27.1.2 virtual activemq::commands::ActiveMQMessage::~ActiveMQMessage ( ) throw () [inline, virtual]

6.27.2 Member Function Documentation

6.27.2.1 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

Implements **cms::Message** (p. 1430).

6.27.2.2 virtual ActiveMQMessage* activemq::commands::ActiveMQMessage::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1417).

6.27.2.3 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. 1417).

6.27.2.4 virtual bool activemq::commands::ActiveMQMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230).

6.27.2.5 virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1419).

6.27.2.6 virtual std::string activemq::commands::ActiveMQMessage::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1424).

6.27.3 Field Documentation

6.27.3.1 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**

6.28 activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 225).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQMessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.28.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 225).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.28.2 Constructor & Destructor Documentation

6.28.2.1 activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller ( ) [inline]

6.28.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::∼ActiveMQMessageMarshaller ( ) [inline, virtual]

6.28.3 Member Function Documentation

6.28.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.28.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.28.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.28.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.28.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.28.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.28.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQMessageMarshaller.h**

6.29 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** () throw ()

- virtual void **acknowledge** () **const**
- virtual void **onSend** ()

  *Allows derived **Message** (p. 1412) classes to perform tasks before a message is sent.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** ()
- virtual void **clearProperties** ()
- virtual std::vector< std::string > **getPropertyNames** () **const**
- virtual bool **propertyExists** (**const** std::string &name) **const**
- virtual bool **getBooleanProperty** (**const** std::string &name) **const**
- virtual unsigned char **getByteProperty** (**const** std::string &name) **const**
- virtual double **getDoubleProperty** (**const** std::string &name) **const**
- virtual float **getFloatProperty** (**const** std::string &name) **const**
- virtual int **getIntProperty** (**const** std::string &name) **const**
- virtual long long **getLongProperty** (**const** std::string &name) **const**
- virtual short **getShortProperty** (**const** std::string &name) **const**
- virtual std::string **getStringProperty** (**const** std::string &name) **const**
- virtual void **setBooleanProperty** (**const** std::string &name, bool value)
- virtual void **setByteProperty** (**const** std::string &name, unsigned char value)
- virtual void **setDoubleProperty** (**const** std::string &name, double value)
- virtual void **setFloatProperty** (**const** std::string &name, float value)
- virtual void **setIntProperty** (**const** std::string &name, int value)
- virtual void **setLongProperty** (**const** std::string &name, long long value)
- virtual void **setShortProperty** (**const** std::string &name, short value)
- virtual void **setStringProperty** (**const** std::string &name, **const** std::string &value)
- virtual std::string **getCMSCorrelationID** () **const**
- virtual void **setCMSCorrelationID** (**const** std::string &**correlationId**)
- virtual int **getCMSDeliveryMode** () **const**
- virtual void **setCMSDeliveryMode** (int mode)
- virtual **const cms::Destination** * **getCMSDestination** () **const**
- virtual void **setCMSDestination** (**const cms::Destination** ***destination**)
- virtual long long **getCMSExpiration** () **const**
- virtual void **setCMSExpiration** (long long expireTime)
- virtual std::string **getCMSMessageID** () **const**
- virtual void **setCMSMessageID** (**const** std::string &id AMQCPP_UNUSED)
- virtual int **getCMSPriority** () **const**
- virtual void **setCMSPriority** (int **priority**)
- virtual bool **getCMSRedelivered** () **const**
- virtual void **setCMSRedelivered** (bool redelivered AMQCPP_UNUSED)
- virtual **const cms::Destination** * **getCMSReplyTo** () **const**
- virtual void **setCMSReplyTo** (**const cms::Destination** ***destination**)
- virtual long long **getCMSTimestamp** () **const**
- virtual void **setCMSTimestamp** (long long timeStamp)
- virtual std::string **getCMSType** () **const**
- virtual void **setCMSType** (**const** std::string &**type**)

Protected Member Functions

- void **failIfWriteOnlyBody** () **const**
- void **failIfReadOnlyBody** () **const**
- void **failIfReadOnlyProperties** () **const**

template<typename T> class activemq::commands::ActiveMQMessageTemplate< T >

6.29.1 Constructor & Destructor Documentation

6.29.1.1 template<typename T> activemq::commands::ActiveMQMessageTemplate< T >::ActiveMQMessageTemplate ( ) [inline]

6.29.1.2 template<typename T> virtual activemq::commands::ActiveMQMessageTemplate< T >::~ActiveMQMessageTemplate ( ) throw () [inline, virtual]

6.29.2 Member Function Documentation

6.29.2.1 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::acknowledge ( ) const [inline, virtual]

6.29.2.2 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearBody ( ) [inline, virtual]

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::ActiveMQStreamMessage** (p. 280), **activemq::commands::ActiveMQMapMessage** (p. 210), and **activemq::commands::ActiveMQTextMessage** (p. 315).

6.29.2.3 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearProperties ( ) [inline, virtual]

6.29.2.4 template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::equals (const DataStructure * *value*) const [inline, virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::Message** (p. 1417).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 132), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::ActiveMQBlobMessage** (p. 124), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 224).

6.29.2.5 template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::failIfReadOnlyBody ( ) const [inline, protected]

6.29.2.6 template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::failIfReadOnlyProperties ( ) const [inline, protected]

6.29.2.7 template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::failIfWriteOnlyBody ( ) const [inline, protected]

6.29.2.8 template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::getBooleanProperty (const std::string & *name*) const [inline, virtual]

6.29.2.9 template<typename T> virtual unsigned char activemq::commands::ActiveMQMessageTemplate< T >::getByteProperty (const std::string & *name*) const [inline, virtual]

6.29.2.10 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSCorrelationID ( ) const [inline, virtual]

6.29.2.11 template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getCMSDeliveryMode ( ) const [inline, virtual]

6.29.2.12 template<typename T> virtual const cms::Destination* activemq::commands::ActiveMQMessageTemplate< T >::getCMSDestination ( ) const [inline, virtual]

6.29.2.13 template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getCMSExpiration ( ) const [inline, virtual]

6.29.2.14 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSMessageID ( ) const [inline, virtual]

6.29.2.15 template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getCMSPriority ( ) const [inline, virtual]

6.29.2.16 template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::getCMSRedelivered ( ) const [inline, virtual]

6.29.2.17 template<typename T> virtual const cms::Destination* activemq::commands::ActiveMQMessageTemplate< T >::getCMSReplyTo ( ) const [inline, virtual]

6.29.2.18 template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getCMSTimestamp ( ) const [inline, virtual]

6.29.2.19 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSType ( ) const [inline, virtual]

6.29.2.20 template<typename T> virtual double activemq::commands::ActiveMQMessageTemplate< T >::getDoubleProperty (const std::string & *name*) const [inline, virtual]

6.29.2.21 template<typename T> virtual float activemq::commands::ActiveMQMessageTemplate< T >::getFloatProperty (const std::string & *name*) const [inline, virtual]

6.29.2.22 template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getIntProperty (const std::string & *name*) const [inline, virtual]

6.29.2.23 template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getLongProperty (const std::string & *name*) const [inline, virtual]

6.29.2.24 template<typename T> virtual std::vector<std::string> activemq::commands::ActiveMQMessageTemplate< T >::getPropertyNames ( ) const [inline, virtual]

6.29.2.25 template<typename T> virtual short activemq::commands::ActiveMQMessageTemplate< T >::getShortProperty (const std::string & *name*) const [inline, virtual]

6.29.2.26 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getStringProperty (const std::string & *name*) const [inline, virtual]

6.29.2.27 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::onSend ( ) [inline, virtual]

Allows derived **Message** (p. 1412) classes to perform tasks before a message is sent.

Reimplemented from **activemq::commands::Message** (p. 1422).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 133), and **activemq::commands::-ActiveMQStreamMessage** (p. 282).

6.29.2.28 template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::propertyExists (const std::string & *name*) const [inline, virtual]

6.29.2.29 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setBooleanProperty (const std::string & *name*, bool *value*) [inline, virtual]

6.29.2.30 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setByteProperty (const std::string & *name*, unsigned char *value*) [inline, virtual]

6.29.2.31 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSCorrelationID (const std::string & *correlationId*) [inline, virtual]

6.29.2.32 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSDeliveryMode (int *mode*) [inline, virtual]

6.29.2.33 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSDestination (const cms::Destination * *destination*) [inline, virtual]

6.29.2.34 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSExpiration (long long *expireTime*) [inline, virtual]

6.29.2.35 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSMessageID (const std::string &id *AMQCPP_UNUSED*) [inline, virtual]

6.29.2.36 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSPriority (int *priority*) [inline, virtual]

6.29.2.37 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSRedelivered (bool redelivered *AMQCPP_UNUSED*) [inline, virtual]

6.29.2.38 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSReplyTo (const cms::Destination * *destination*) [inline, virtual]

6.29.2.39 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSTimestamp (long long *timeStamp*) [inline, virtual]

6.29.2.40 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSType (const std::string & *type*) [inline, virtual]

6.29.2.41 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setDoubleProperty (const std::string & *name*, double *value*) [inline, virtual]

6.29.2.42 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setFloatProperty (const std::string & *name*, float *value*) [inline, virtual]

6.29.2.43 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setIntProperty (const std::string & *name*, int *value*) [inline, virtual]

6.29.2.44 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setLongProperty (const std::string & *name*, long long *value*) [inline, virtual]

6.29.2.45 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setShortProperty (const std::string & *name*, short *value*) [inline, virtual]

6.29.2.46 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setStringProperty (const std::string & *name*, const std::string & *value*) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessageTemplate.h**

6.30 activemq::commands::ActiveMQObjectMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQObjectMessage.h>`

Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual ~**ActiveMQObjectMessage** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQObjectMessage** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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

6.30.1 Constructor & Destructor Documentation

6.30.1.1 activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage ( )

6.30.1.2 virtual activemq::commands::ActiveMQObjectMessage::~ActiveMQObjectMessage ( ) throw () [inline, virtual]

6.30.2 Member Function Documentation

6.30.2.1 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

Implements **cms::Message** (p. 1430).

6.30.2.2 virtual ActiveMQObjectMessage* activemq::commands::ActiveMQObjectMessage::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1417).

6.30.2.3 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. 1417).

6.30.2.4 virtual bool activemq::commands::ActiveMQObjectMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230).

6.30.2.5 virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1419).

6.30.2.6 virtual std::string activemq::commands::ActiveMQObjectMessage::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1424).

6.30.3 Field Documentation

6.30.3.1 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**

6.31 activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 235).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQObjectMessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 235).

NOTE!: This file is 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::generated::ActiveMQObjectMessageMarshaller::-ActiveMQObjectMessageMarshaller ( ) [inline]

6.31.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQObject-MessageMarshaller::~ActiveMQObjectMessageMarshaller ( ) [inline, virtual]

6.31.3 Member Function Documentation

6.31.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal-::generated::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.31.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessage-Marshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.31.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessage-Marshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.31.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.31.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.31.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.31.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQObjectMessageMarshaller.h**

6.32 activemq::core::ActiveMQProducer Class Reference

`#include <src/main/activemq/core/ActiveMQProducer.h>`

Inheritance diagram for activemq::core::ActiveMQProducer:

Public Member Functions

- **ActiveMQProducer** (**ActiveMQSession** *session, **const Pointer**< **commands::ProducerId** > &producerId, **const Pointer**< **commands::ActiveMQDestination** > &destination, long long sendTimeout)

  *Constructor, creates an instance of an **ActiveMQProducer** (p. 238).*
- virtual ~**ActiveMQProducer** () throw ()
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- virtual void **send** (**cms::Message** *message)

  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)

  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)

  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)

  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)

  Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () **const**

  Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value)

  Sets if Message Ids are disabled for this Producer.
- virtual bool **getDisableMessageID** () **const**

  Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value)

  Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () **const**

  Gets if Message Time Stamps are disabled for this Producer.
- virtual void **setPriority** (int priority)

  Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () **const**

  Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time)

  Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () **const**

  Gets the Time to Live that this producer sends messages with.
- virtual void **setSendTimeout** (long long time)

  Sets the Send Timeout that this Producers sends messages with.
- virtual long long **getSendTimeout** () **const**

  Gets the Send Timeout that this producer sends messages with.
- bool **isClosed** () **const**
- **const Pointer**
  < commands::ProducerInfo > & **getProducerInfo** () **const**

  Retries this object ProducerInfo pointer.
- **const 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.
- void **dispose** ()

  Performs Producer object cleanup but doesn't attempt to send the Remove command to the broker.

6.32.1 Constructor & Destructor Documentation

6.32.1.1 activemq::core::ActiveMQProducer::ActiveMQProducer (ActiveMQSession * *session*, const Pointer< commands::ProducerId > & *producerId*, const Pointer< commands::ActiveMQDestination > & *destination*, long long *sendTimeout*)

Constructor, creates an instance of an **ActiveMQProducer** (p. 238).

Parameters

session	The Session which is the parent of this Producer.
producerId	Pointer to a ProducerId object 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.
sendTimeout	The configured send timeout for this Producer.

6.32.1.2 virtual activemq::core::ActiveMQProducer::~ActiveMQProducer ( ) throw () [virtual]

6.32.2 Member Function Documentation

6.32.2.1 virtual void activemq::core::ActiveMQProducer::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

CMSException	- If an error occurs while the resource is being closed.

Implements **cms::Closeable** (p. 633).

6.32.2.2 void activemq::core::ActiveMQProducer::dispose ( )

Performs Producer object cleanup but doesn't attempt to send the Remove command to the broker.

Called when the parent resource if closed first to avoid the message send and avoid any exceptions that might be thrown from an attempt to send a remove command to a failed transport.

6.32.2.3 virtual int activemq::core::ActiveMQProducer::getDeliveryMode ( ) const [inline, virtual]

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Implements **cms::MessageProducer** (p. 1493).

6.32.2.4 virtual bool activemq::core::ActiveMQProducer::getDisableMessageID ( ) const [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. 1493).

6.32.2.5 virtual bool activemq::core::ActiveMQProducer::getDisableMessageTimeStamp ( ) const [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. 1493).

6.32.2.6 virtual int activemq::core::ActiveMQProducer::getPriority ( ) const [inline, virtual]

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Implements **cms::MessageProducer** (p. 1494).

6.32.2.7 const Pointer<commands::ProducerId>& activemq::core::ActiveMQProducer::getProducerId ( ) const [inline]

Retries this object ProducerId or NULL if closed.

Returns

ProducerId Reference

6.32.2.8 const Pointer<commands::ProducerInfo>& activemq::core::ActiveMQProducer::getProducerInfo ( ) const [inline]

Retries this object ProducerInfo pointer.

Returns

ProducerInfo Reference

6.32.2.9 virtual long long activemq::core::ActiveMQProducer::getSendTimeout ( ) const [inline, virtual]

Gets the Send Timeout that this producer sends messages with.

Returns

The default send timeout value in milliseconds.

6.32.2.10 virtual long long activemq::core::ActiveMQProducer::getTimeToLive ( ) const [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. 1494).

6.32.2.11 bool activemq::core::ActiveMQProducer::isClosed ( ) const [inline]

Returns

true if this Producer has been closed.

6.32.2.12 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.

6.32.2.13 virtual void activemq::core::ActiveMQProducer::send (cms::Message * *message*) [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.
UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1494).

6.32.2.14 virtual void activemq::core::ActiveMQProducer::send (cms::Message * *message,* int *deliveryMode,* int *priority,* long long *timeToLive*) [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.
UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1495).

6.32.2.15 virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * *destination*, cms::Message * *message*) [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.
UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1495).

6.32.2.16 virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * *destination*, cms::Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) [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.
UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1496).

6.32.2.17 virtual void activemq::core::ActiveMQProducer::setDeliveryMode (int *mode*) [inline, virtual]

Sets the delivery mode for this Producer.

Parameters

mode	- The DeliveryMode to use for Message sends.

Implements **cms::MessageProducer** (p. 1496).

6.32.2.18 virtual void activemq::core::ActiveMQProducer::setDisableMessageID (bool *value*) [inline, virtual]

Sets if Message Ids are disabled for this Producer.

Parameters

value	- boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 1496).

6.32.2.19 virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp (bool *value*) [inline, virtual]

Sets if Message Time Stamps are disabled for this Producer.

Parameters

value	- boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 1497).

6.32.2.20 virtual void activemq::core::ActiveMQProducer::setPriority (int *priority*) [inline, virtual]

Sets the Priority that this Producers sends messages at.

Parameters

priority	int value for Priority level

Implements **cms::MessageProducer** (p. 1497).

6.32.2.21 virtual void activemq::core::ActiveMQProducer::setSendTimeout (long long *time*) [inline, virtual]

Sets the Send Timeout that this Producers sends messages with.

Parameters

time	The new default send timeout value in milliseconds.

6.32.2.22 virtual void activemq::core::ActiveMQProducer::setTimeToLive (long long *time*) [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. 1497).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQProducer.h**

6.33 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 1705) object.

`#include <src/main/activemq/util/ActiveMQProperties.h>`

Inheritance diagram for activemq::util::ActiveMQProperties:

Public Member Functions

- **ActiveMQProperties** ()
- virtual ∼**ActiveMQProperties** () throw ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual **const** **decaf::util::Properties** & **getProperties** () **const**
- virtual void **setProperties** (**decaf::util::Properties** &props)
- virtual int **size** () **const**

  Returns the current count of all the Properties that are currently stored in the Properties object.
- 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 std::string **remove** (**const** std::string &name)

  Removes the property with the given name.
- virtual std::vector< std::string > **propertyNames** () **const**

  Returns a vector containing all the names of the properties currently stored in the Properties object.
- 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)
- 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.

6.33.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 1705) object.

Since

2.0

6.33.2 Constructor & Destructor Documentation

6.33.2.1 activemq::util::ActiveMQProperties::ActiveMQProperties ( )

6.33.2.2 virtual activemq::util::ActiveMQProperties::~ActiveMQProperties ( ) throw () [virtual]

6.33.3 Member Function Documentation

6.33.3.1 virtual void activemq::util::ActiveMQProperties::clear ( ) [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 645).

6.33.3.2 virtual CMSProperties* activemq::util::ActiveMQProperties::clone ( ) const [virtual]

Clones this object.

Returns

a replica of this object.

Implements **cms::CMSProperties** (p. 645).

6.33.3.3 virtual void activemq::util::ActiveMQProperties::copy (const CMSProperties * *source*) [virtual]

6.33.3.4 virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ( ) [inline, virtual]

6.33.3.5 virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ( ) const [inline, virtual]

6.33.3.6 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. 645).

6.33.3.7 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. 646).

6.33.3.8 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	the name of the property to check

Returns

true if property exists, false otherwise.

Implements **cms::CMSProperties** (p. 646).

6.33.3.9 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. 646).

6.33.3.10 virtual std::vector<std::string> activemq::util::ActiveMQProperties::propertyNames ( ) const [inline, virtual]

Returns a vector containing all the names of the properties currently stored in the Properties object.

Returns

an STL std::vector<std::string> with all the currently stored property names.

Implements **cms::CMSProperties** (p. 646).

6.33.3.11 virtual std::string activemq::util::ActiveMQProperties::remove (const std::string & *name*) [inline, virtual]

Removes the property with the given name.

If the property existed in the collection then it is removed and returned, otherwise an empty string is returned.

Parameters

name	the name of the property to be removed.

Returns

the value that was removed from the Properties, or empty string.

Implements **cms::CMSProperties** (p. 647).

6.33.3.12 virtual void activemq::util::ActiveMQProperties::setProperties (decaf::util::Properties & *props*) [inline, virtual]

6.33.3.13 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. 647).

6.33.3.14 virtual int activemq::util::ActiveMQProperties::size ( ) const [inline, virtual]

Returns the current count of all the Properties that are currently stored in the Properties object.

Returns

the number of properties currently stored.

Implements **cms::CMSProperties** (p. 647).

6.33.3.15 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. 647).

6.33.3.16 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. 647).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**ActiveMQProperties.h**

6.34 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** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQQueue** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual **const cms::Destination** ∗ **getCMSDestination** () **const**
- virtual
  cms::Destination::DestinationType getDestinationType () **const**

  Returns the Type of Destination that this object represents.
- 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 bool **equals** (**const cms::Destination** &other) **const**

  Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getQueueName** () **const**

  Gets the name of this queue.

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQQUEUE** = 100

6.34.1 Constructor & Destructor Documentation

6.34.1.1 activemq::commands::ActiveMQQueue::ActiveMQQueue ( )

6.34.1.2 activemq::commands::ActiveMQQueue::ActiveMQQueue (const std::string & *name*)

6.34.1.3 virtual activemq::commands::ActiveMQQueue::∼ActiveMQQueue ( ) throw () [virtual]

6.34.2 Member Function Documentation

6.34.2.1 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. 937).

6.34.2.2 virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::cloneDataStructure ( ) const [virtual]

Clone this obbject 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::ActiveMQDestination** (p. 193).

6.34.2.3 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. 938).

6.34.2.4 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

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 193).

6.34.2.5 virtual bool activemq::commands::ActiveMQQueue::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 194).

6.34.2.6 virtual bool activemq::commands::ActiveMQQueue::equals (const cms::Destination & *other*) const [virtual]

Compares two Destination instances to determine if they represent the same logic Destination.

Parameters

other	The other destination to compare this one to.

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 938).

6.34.2.7 virtual const cms::Destination* activemq::commands::ActiveMQQueue::getCMSDestination ( ) const [inline, virtual]

Returns

the **cms::Destination** (p. 936) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 195).

6.34.2.8 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

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 938).

6.34.2.9 virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 195).

6.34.2.10 virtual cms::Destination::DestinationType activemq::commands::ActiveMQQueue::getDestinationType ( ) const [inline, virtual]

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 195).

References cms::Destination::QUEUE.

6.34.2.11 virtual std::string activemq::commands::ActiveMQQueue::getQueueName ( ) const [inline, virtual]

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Queue** (p. 1722).

6.34.2.12 virtual std::string activemq::commands::ActiveMQQueue::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 198).

6.34.3 Field Documentation

6.34.3.1 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**

6.35 activemq::core::ActiveMQQueueBrowser Class Reference

`#include <src/main/activemq/core/ActiveMQQueueBrowser.h>`

Inheritance diagram for activemq::core::ActiveMQQueueBrowser:

Public Member Functions

- **ActiveMQQueueBrowser** (**ActiveMQSession** *session, **const Pointer**< **commands::ConsumerId** > &consumerId, **const Pointer**< **commands::ActiveMQDestination** > &destination, **const** std::string &selector, bool dispatchAsync)
- virtual ~**ActiveMQQueueBrowser** () throw ()
- virtual **const cms::Queue** * **getQueue** () **const**
- virtual std::string **getMessageSelector** () **const**
- virtual **cms::MessageEnumeration** * **getEnumeration** ()

  Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.
- virtual void **close** ()

Closes this object and deallocates the appropriate resources.

- virtual bool **hasMoreMessages** ()

  Returns true if there are more Message in the Browser that can be retrieved via the `nextMessage` *method.*

- virtual **cms::Message** * **nextMessage** ()

  Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

Friends

- class **Browser**

6.35.1 Constructor & Destructor Documentation

6.35.1.1 activemq::core::ActiveMQQueueBrowser::ActiveMQQueueBrowser (ActiveMQSession * *session,* const Pointer< commands::ConsumerId > & *consumerId,* const Pointer< commands::ActiveMQDestination > & *destination,* const std::string & *selector,* bool *dispatchAsync*)

6.35.1.2 virtual activemq::core::ActiveMQQueueBrowser::~ActiveMQQueueBrowser ( ) throw () [virtual]

6.35.2 Member Function Documentation

6.35.2.1 virtual void activemq::core::ActiveMQQueueBrowser::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

CMSException	- If an error occurs while the resource is being closed.

Implements **cms::Closeable** (p. 633).

6.35.2.2 virtual cms::MessageEnumeration* activemq::core::ActiveMQQueueBrowser::getEnumeration ( ) [virtual]

Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

Returns

a pointer to a Queue Enumeration, this Pointer is owned by the QueueBrowser and should not be deleted by the client.

Exceptions

CMSException	if an internal error occurs.

Implements **cms::QueueBrowser** (p. 1727).

6.35.2.3 virtual std::string activemq::core::ActiveMQQueueBrowser::getMessageSelector ( ) const [virtual]

Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions

CMSException	if an internal error occurs.

Implements **cms::QueueBrowser** (p. 1727).

6.35.2.4 virtual const cms::Queue* activemq::core::ActiveMQQueueBrowser::getQueue ( ) const [virtual]

Returns

the Queue that this browser is listening on.

Exceptions

CMSException	if an internal error occurs.

Implements **cms::QueueBrowser** (p. 1727).

6.35.2.5 virtual bool activemq::core::ActiveMQQueueBrowser::hasMoreMessages ( ) [virtual]

Returns true if there are more Message in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

Returns

true if more Message's are available in the Browser.

Implements **cms::MessageEnumeration** (p. 1476).

6.35.2.6 virtual cms::Message* activemq::core::ActiveMQQueueBrowser::nextMessage ( ) [virtual]

Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

If a Message object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

Returns

The next Message in the Queue.

Exceptions

CMSException	if no more Message's currently in the Queue.

Implements **cms::MessageEnumeration** (p. 1477).

6.35.3 Friends And Related Function Documentation

6.35.3.1 friend class Browser [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQQueueBrowser.h**

6.36 activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 255).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQQueueMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual ~**ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.36.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 255).

NOTE!: This file is 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::generated::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller ( ) [inline]

6.36.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::~ActiveMQQueueMarshaller ( ) [inline, virtual]

6.36.3 Member Function Documentation

6.36.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.36.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.36.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

6.36.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

6.36.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

6.36.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202).

6.36.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQQueueMarshaller.h**

6.37 activemq::core::ActiveMQSession Class Reference

`#include <src/main/activemq/core/ActiveMQSession.h>`

Inheritance diagram for activemq::core::ActiveMQSession:

Public Member Functions

- **ActiveMQSession** (**ActiveMQConnection** *__connection__, **const Pointer**< **commands::SessionId** > &id, **cms::Session::AcknowledgeMode ackMode**, **const decaf::util::Properties** &properties)
- virtual ~**ActiveMQSession** () throw ()
- virtual void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)

  Redispatches the given set of unconsumed messages to the consumers.
- virtual void **start** ()

  Stops asynchronous message delivery.
- virtual void **stop** ()

  Starts asynchronous message delivery.
- bool **isStarted** () **const**

  Indicates whether or not the session is currently in the started state.
- virtual bool **isAutoAcknowledge** () **const**
- virtual bool **isDupsOkAcknowledge** () **const**
- virtual bool **isClientAcknowledge** () **const**
- virtual 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** ()

Closes this session as well as any active child consumers or producers.

- virtual void **commit** ()

  Commits all messages done in this transaction and releases any locks currently held.

- virtual void **rollback** ()

  Rolls back all messages done in this transaction and releases any locks currently held.

- virtual void **recover** ()

  Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

- virtual **cms::MessageConsumer** * **createConsumer** (**const cms::Destination** *destination)

  Creates a MessageConsumer for the specified destination.

- virtual **cms::MessageConsumer** * **createConsumer** (**const cms::Destination** *destination, **const** std::string &selector)

  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)

  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)

  Creates a durable subscriber to the specified topic, using a Message selector.

- virtual **cms::MessageProducer** * **createProducer** (**const cms::Destination** *destination)

  Creates a MessageProducer to send messages to the specified destination.

- virtual **cms::QueueBrowser** * **createBrowser** (**const cms::Queue** *queue)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::QueueBrowser** * **createBrowser** (**const cms::Queue** *queue, **const** std::string &selector)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::Queue** * **createQueue** (**const** std::string &queueName)

  Creates a queue identity given a Queue name.

- virtual **cms::Topic** * **createTopic** (**const** std::string &topicName)

  Creates a topic identity given a Queue name.

- virtual **cms::TemporaryQueue** * **createTemporaryQueue** ()

  Creates a TemporaryQueue object.

- virtual **cms::TemporaryTopic** * **createTemporaryTopic** ()

  Creates a TemporaryTopic object.

- virtual **cms::Message** * **createMessage** ()

  Creates a new Message.

- virtual **cms::BytesMessage** * **createBytesMessage** ()

  Creates a BytesMessage.

- virtual **cms::BytesMessage** * **createBytesMessage** (**const** unsigned char *bytes, int bytesSize)

  Creates a BytesMessage and sets the payload to the passed value.

- virtual **cms::StreamMessage** * **createStreamMessage** ()

  Creates a new StreamMessage.

- virtual **cms::TextMessage** * **createTextMessage** ()

  Creates a new TextMessage.

- virtual **cms::TextMessage** * **createTextMessage** (**const** std::string &text)

  Creates a new TextMessage and set the text to the value given.

- virtual **cms::MapMessage** * **createMapMessage** ()

  Creates a new MapMessage.

- virtual **cms::Session::AcknowledgeMode getAcknowledgeMode** () **const**

  Returns the acknowledgment mode of the session.

- virtual bool **isTransacted** () **const**

  Gets if the Sessions is a Transacted Session.

- virtual void **unsubscribe** (**const** std::string &name)

  Unsubscribes a durable subscription that has been created by a client.
- void **send** (**cms::Message** ∗message, **ActiveMQProducer** ∗producer, **util::Usage** ∗usage)

  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. 145) that is associated with this session.*
- **Pointer**< **threads::Scheduler** > **getScheduler** () **const**

  Gets a Pointer to this Session's Scheduler instance.
- 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)

  Sends a Command to the broker without requesting any Response be returned.
- **Pointer**< **commands::Response** > **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0)

  Sends a synchronous request and returns the response from the broker.
- void **addConsumer** (**ActiveMQConsumer** ∗consumer)

  Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.
- void **removeConsumer** (**const Pointer**< **commands::ConsumerId** > &consumerId)

  Dispose of a MessageConsumer from this session.
- void **addProducer** (**ActiveMQProducer** ∗producer)

  Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.
- void **removeProducer** (**ActiveMQProducer** ∗producer)

  Dispose of a MessageProducer from this session.
- virtual void **doStartTransaction** ()

  Starts if not already start a Transaction for this Session.
- **Pointer**
  < **ActiveMQTransactionContext** > **getTransactionContext** ()

  Gets the Pointer to this Session's TransactionContext.
- 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 executer and ensure all messages are dispatched.
- **Pointer**< **commands::ConsumerId** > **getNextConsumerId** ()

  Get the Next available Consumer Id.
- **Pointer**< **commands::ProducerId** > **getNextProducerId** ()

  Get the Next available Producer Id.

- void **doClose** ()

  Performs the actual Session close operations.
- void **dispose** ()

  Cleans up the Session object's resources without attempting to send the Remove command to the broker, this can be called from ***ActiveMQConnection*** *(*p. *145) when it knows that the transport is down and the doClose method would throw an exception when it attempt to send the Remove Command.*

Protected Attributes

- SessionConfig ∗ **config**
- **Pointer**< **commands::SessionInfo** > **sessionInfo**

  SessionInfo for this Session.
- **Pointer**
  < **ActiveMQTransactionContext** > **transaction**

  Transaction Management object.
- **ActiveMQConnection** ∗ **connection**

  Connection.
- **ConsumersMap consumers**

  Map of consumers.
- **AtomicBoolean closed**

  Indicates that this connection has been closed, it is no longer usable after this becomes true.
- std::auto_ptr
  < **ActiveMQSessionExecutor** > **executor**

  Sends incoming messages to the registered consumers.
- **cms::Session::AcknowledgeMode ackMode**

  This Sessions Acknowledgment mode.
- **util::LongSequenceGenerator producerIds**

  Next available Producer Id.
- **util::LongSequenceGenerator producerSequenceIds**

  Next available Producer Sequence Id.
- **util::LongSequenceGenerator consumerIds**

  Next available Consumer Id.
- long long **lastDeliveredSequenceId**

  Last Delivered Sequence Id.

Friends

- class **ActiveMQSessionExecutor**

6.37.1 Constructor & Destructor Documentation

6.37.1.1 activemq::core::ActiveMQSession::ActiveMQSession (ActiveMQConnection ∗ *connection,* const Pointer< commands::SessionId > & *id,* cms::Session::AcknowledgeMode *ackMode,* const decaf::util::Properties & *properties*)

6.37.1.2 virtual activemq::core::ActiveMQSession::∼ActiveMQSession ( ) throw () [virtual]

6.37.2 Member Function Documentation

6.37.2.1 void activemq::core::ActiveMQSession::acknowledge ( )

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

6.37.2.2 void activemq::core::ActiveMQSession::addConsumer (ActiveMQConsumer * *consumer*)

Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

Parameters

consumer	The **ActiveMQConsumer** (p. 181) instance to add to this session.

Exceptions

ActiveMQException	if an internal error occurs.

6.37.2.3 void activemq::core::ActiveMQSession::addProducer (ActiveMQProducer * *producer*)

Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

Parameters

consumer	The **ActiveMQProducer** (p. 238) instance to add to this session.

Exceptions

ActiveMQException	if an internal error occurs.

6.37.2.4 void activemq::core::ActiveMQSession::clearMessagesInProgress ( )

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

6.37.2.5 virtual void activemq::core::ActiveMQSession::close ( ) [virtual]

Closes this session as well as any active child consumers or producers.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1833).

6.37.2.6 virtual void activemq::core::ActiveMQSession::commit ( ) [virtual]

Commits 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. 1834).

Reimplemented in **activemq::core::ActiveMQXASession** (p. 339).

6.37.2.7 virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * *queue*) [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.

Implements **cms::Session** (p. 1834).

6.37.2.8 virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * *queue,* const std::string & *selector*) [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

CMSException	- If an internal error occurs.
InvalidDestinationException	- if the destination given is invalid.

Implements **cms::Session** (p. 1834).

6.37.2.9 virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage ( ) [virtual]

Creates a BytesMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1835).

6.37.2.10 virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage (const unsigned char * *bytes*, int *bytesSize*) [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

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1835).

6.37.2.11 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * *destination*) [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. 1835).

6.37.2.12 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * *destination*, const std::string & *selector*) [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. 1836).

6.37.2.13 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * *destination,* const std::string & *selector,* bool *noLocal*) [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. 1836).

6.37.2.14 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer (const cms::Topic * *destination,* const std::string & *name,* const std::string & *selector,* bool *noLocal* = false) [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. 1837).

6.37.2.15 virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage ( ) [virtual]

Creates a new MapMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1837).

6.37.2.16 virtual cms::Message* activemq::core::ActiveMQSession::createMessage ( ) [virtual]

Creates a new Message.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1837).

6.37.2.17 virtual cms::MessageProducer* activemq::core::ActiveMQSession::createProducer (const cms::Destination * *destination*) [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. 1838).

6.37.2.18 virtual cms::Queue* activemq::core::ActiveMQSession::createQueue (const std::string & *queueName*) [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

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1838).

6.37.2.19 virtual cms::StreamMessage* activemq::core::ActiveMQSession::createStreamMessage ( ) [virtual]

Creates a new StreamMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1838).

6.37.2.20 virtual cms::TemporaryQueue* activemq::core::ActiveMQSession::createTemporaryQueue ( ) [virtual]

Creates a TemporaryQueue object.

Returns

new TemporaryQueue pointer that is owned by the caller.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1839).

6.37.2.21 virtual cms::TemporaryTopic* activemq::core::ActiveMQSession::createTemporaryTopic ( ) [virtual]

Creates a TemporaryTopic object.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1839).

6.37.2.22 virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage ( ) [virtual]

Creates a new TextMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1839).

6.37.2.23 virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage (const std::string & *text*) [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. 1839).

6.37.2.24 virtual cms::Topic* activemq::core::ActiveMQSession::createTopic (const std::string & *topicName*) [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. 1840).

6.37.2.25 void activemq::core::ActiveMQSession::deliverAcks ( )

Request that this Session inform all of its consumers to deliver their pending acks.

6.37.2.26 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. 956).

6.37.2.27 void activemq::core::ActiveMQSession::dispose ( )

Cleans up the Session object's resources without attempting to send the Remove command to the broker, this can be called from **ActiveMQConnection** (p. 145) when it knows that the transport is down and the doClose method would throw an exception when it attempt to send the Remove Command.

6.37.2.28 void activemq::core::ActiveMQSession::doClose ( )

Performs the actual Session close operations.

This method is meant for use by **ActiveMQConnection** (p. 145), the connection object calls this when it has been closed to skip some of the extraneous processing done by the client level close method.

6.37.2.29 virtual void activemq::core::ActiveMQSession::doStartTransaction ( ) [virtual]

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.

Reimplemented in **activemq::core::ActiveMQXASession** (p. 339).

6.37.2.30 void activemq::core::ActiveMQSession::fire (const exceptions::ActiveMQException & *ex*)

Fires the given exception to the exception listener of the connection.

6.37.2.31 virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode ( ) const [virtual]

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1840).

6.37.2.32 ActiveMQConnection* activemq::core::ActiveMQSession::getConnection ( ) const [inline]

Gets the **ActiveMQConnection** (p. 145) that is associated with this session.

6.37.2.33 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. 996) pointer or NULL

6.37.2.34 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.

6.37.2.35 Pointer<commands::ConsumerId> activemq::core::ActiveMQSession::getNextConsumerId ( )

Get the Next available Consumer Id.

Returns

the next id in the sequence.

6.37.2.36 Pointer<commands::ProducerId> activemq::core::ActiveMQSession::getNextProducerId ( )

Get the Next available Producer Id.

Returns

the next id in the sequence.

6.37.2.37 Pointer<threads::Scheduler> activemq::core::ActiveMQSession::getScheduler ( ) const

Gets a Pointer to this Session's Scheduler instance.

6.37.2.38 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

6.37.2.39 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

6.37.2.40 Pointer<ActiveMQTransactionContext> activemq::core::ActiveMQSession::getTransactionContext ( ) [inline]

Gets the Pointer to this Session's TransactionContext.

Returns

a Pointer to this Session's TransactionContext

6.37.2.41 virtual bool activemq::core::ActiveMQSession::isAutoAcknowledge ( ) const [inline, virtual]

Reimplemented in **activemq::core::ActiveMQXASession** (p. 339).

References cms::Session::AUTO_ACKNOWLEDGE.

6.37.2.42 virtual bool activemq::core::ActiveMQSession::isClientAcknowledge ( ) const [inline, virtual]

References cms::Session::CLIENT_ACKNOWLEDGE.

6.37.2.43 virtual bool activemq::core::ActiveMQSession::isDupsOkAcknowledge ( ) const [inline, virtual]

References cms::Session::DUPS_OK_ACKNOWLEDGE.

6.37.2.44 virtual bool activemq::core::ActiveMQSession::isIndividualAcknowledge ( ) const [inline, virtual]

References cms::Session::INDIVIDUAL_ACKNOWLEDGE.

6.37.2.45 bool activemq::core::ActiveMQSession::isStarted ( ) const

Indicates whether or not the session is currently in the started state.

6.37.2.46 virtual bool activemq::core::ActiveMQSession::isTransacted ( ) const [virtual]

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1840).

Reimplemented in **activemq::core::ActiveMQXASession** (p. 340).

6.37.2.47 void activemq::core::ActiveMQSession::oneway (Pointer< commands::Command > *command*)

Sends a Command to the broker without requesting any Response be returned.

Parameters

command	The message to send to the Broker.

Exceptions

ActiveMQException	if not currently connected, or if the operation fails for any reason.

6.37.2.48 virtual void activemq::core::ActiveMQSession::recover ( ) [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. 1840).

6.37.2.49 virtual void activemq::core::ActiveMQSession::redispatch (MessageDispatchChannel & *unconsumedMessages*) [virtual]

Redispatches the given set of unconsumed messages to the consumers.

Parameters

unconsumed-Messages	- unconsumed messages to be redelivered.

6.37.2.50 void activemq::core::ActiveMQSession::removeConsumer (const Pointer< commands::ConsumerId > & *consumerId*)

Dispose of a MessageConsumer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

consumerId	The ConsumerId of the MessageConsumer to remove from this Session.

Exceptions

ActiveMQException	if an internal error occurs.

6.37.2.51 void activemq::core::ActiveMQSession::removeProducer (ActiveMQProducer * *producer*)

Dispose of a MessageProducer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

producerId	The ProducerId of the MessageProducer to remove from this session.

Exceptions

ActiveMQException	if an internal error occurs.

6.37.2.52 virtual void activemq::core::ActiveMQSession::rollback ( ) [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. 1841).

Reimplemented in **activemq::core::ActiveMQXASession** (p. 340).

6.37.2.53 void activemq::core::ActiveMQSession::send (cms::Message * *message*, ActiveMQProducer * *producer*, util::Usage * *usage*)

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	

6.37.2.54 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.

6.37.2.55 virtual void activemq::core::ActiveMQSession::start ( ) [virtual]

Stops asynchronous message delivery.

Implements **cms::Startable** (p. 1964).

6.37.2.56 virtual void activemq::core::ActiveMQSession::stop ( ) [virtual]

Starts asynchronous message delivery.

Implements **cms::Stoppable** (p. 2017).

6.37.2.57 Pointer<commands::Response> activemq::core::ActiveMQSession::syncRequest (Pointer< commands::Command > *command*, unsigned int *timeout* = 0)

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

Parameters

command	The command to send to the broker.
timeout	The time to wait for a response, default is zero or infinite.

Returns

Pointer to a Response object that the broker has returned for the Command sent.

Exceptions

ActiveMQException	thrown if an error response was received from the broker, or if any other error occurred.

6.37.2.58 virtual void activemq::core::ActiveMQSession::unsubscribe (const std::string & *name*) [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	- If an internal error occurs.

Implements **cms::Session** (p. 1841).

6.37.2.59 void activemq::core::ActiveMQSession::wakeup ( )

Causes the Session to wakeup its executer and ensure all messages are dispatched.

6.37.3 Friends And Related Function Documentation

6.37.3.1 friend class ActiveMQSessionExecutor [friend]

6.37.4 Field Documentation

6.37.4.1 cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::ackMode [protected]

This Sessions Acknowledgment mode.

6.37.4.2 AtomicBoolean activemq::core::ActiveMQSession::closed [protected]

Indicates that this connection has been closed, it is no longer usable after this becomes true.

6.37.4.3 SessionConfig* activemq::core::ActiveMQSession::config [protected]

6.37.4.4 ActiveMQConnection* activemq::core::ActiveMQSession::connection [protected]

Connection.

6.37.4.5 util::LongSequenceGenerator activemq::core::ActiveMQSession::consumerIds [protected]

Next available Consumer Id.

6.37.4.6 ConsumersMap activemq::core::ActiveMQSession::consumers [protected]

Map of consumers.

6.37.4.7 std::auto_ptr<ActiveMQSessionExecutor> activemq::core::ActiveMQSession::executor [protected]

Sends incoming messages to the registered consumers.

6.37.4.8 long long activemq::core::ActiveMQSession::lastDeliveredSequenceId [protected]

Last Delivered Sequence Id.

6.37.4.9 util::LongSequenceGenerator activemq::core::ActiveMQSession::producerIds [protected]

Next available Producer Id.

6.37.4.10 util::LongSequenceGenerator activemq::core::ActiveMQSession::producerSequenceIds [protected]

Next available Producer Sequence Id.

6.37.4.11 Pointer<commands::SessionInfo> activemq::core::ActiveMQSession::sessionInfo [protected]

SessionInfo for this Session.

6.37.4.12 Pointer<ActiveMQTransactionContext> activemq::core::ActiveMQSession::transaction [protected]

Transaction Management object.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQSession.h**

6.38 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. 278) then **clear()** (p. 277).*
- 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 **hasUncomsumedMessages** () **const**
- virtual void **wakeup** ()

  wakeup this executer 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. 1462) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer**
  < **MessageDispatch** > > **getUnconsumedMessages** ()

6.38.1 Detailed Description

Delegate dispatcher for a single session.

Contains a thread to provide for asynchronous dispatching.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor (ActiveMQSession * *session*)

Creates an un-started executor for the given session.

6.38.2.2 virtual activemq::core::ActiveMQSessionExecutor::~ActiveMQSessionExecutor ( ) [virtual]

Calls **stop()** (p. 278) then **clear()** (p. 277).

6.38.3 Member Function Documentation

6.38.3.1 virtual void activemq::core::ActiveMQSessionExecutor::clear ( ) [inline, virtual]

Removes all queued messages and destroys them.

6.38.3.2 virtual void activemq::core::ActiveMQSessionExecutor::clearMessagesInProgress ( ) [inline, virtual]

Removes all messages in the Dispatch Channel so that non are delivered.

6.38.3.3 virtual void activemq::core::ActiveMQSessionExecutor::close ( ) [inline, virtual]

Terminates the dispatching thread.

Once this is called, the executor is no longer usable.

6.38.3.4 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.

6.38.3.5 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.

6.38.3.6 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.

6.38.3.7 virtual bool activemq::core::ActiveMQSessionExecutor::hasUncomsumedMessages ( ) const [inline, virtual]

Returns

true if there are any pending messages in the dispatch channel.

6.38.3.8 virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ( ) [inline, virtual]

Returns

true if there are no messages in the Dispatch Channel.

6.38.3.9 virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ( ) const [inline, virtual]

Returns

true indicates if the executor is started

6.38.3.10 virtual bool activemq::core::ActiveMQSessionExecutor::iterate ( ) [virtual]

Iterates on the **MessageDispatchChannel** (p. 1462) 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. 2073).

6.38.3.11 virtual void activemq::core::ActiveMQSessionExecutor::start ( ) [virtual]

Starts the dispatching.

6.38.3.12 virtual void activemq::core::ActiveMQSessionExecutor::stop ( ) [virtual]

Stops dispatching.

6.38.3.13 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**

6.39 activemq::commands::ActiveMQStreamMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQStreamMessage.h>`

Inheritance diagram for activemq::commands::ActiveMQStreamMessage:

Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual ~**ActiveMQStreamMessage** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQStreamMessage** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()

  *Allows derived **Message** (p. 1412) classes to perform tasks before a message is sent.*
- virtual **cms::StreamMessage** * **clone** () **const**

  Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **clearBody** ()

  Clears out the body of the message.
- virtual void **reset** ()
- virtual bool **readBoolean** () **const**

  Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value)

  Writes a boolean to the Stream message stream as a 1-byte value.
- virtual unsigned char **readByte** () **const**

  Reads a Byte from the Stream message stream.
- virtual void **writeByte** (unsigned char value)

  Writes a byte to the Stream message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) **const**

  Reads a byte array from the Stream message stream.
- virtual void **writeBytes** (**const** std::vector< unsigned char > &value)

  Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char *buffer, int length) **const**

  Reads a portion of the Stream message stream.
- virtual void **writeBytes** (**const** unsigned char *value, int offset, int length)

  Writes a portion of a byte array to the Stream message stream.
- virtual char **readChar** () **const**

  Reads a Char from the Stream message stream.
- virtual void **writeChar** (char value)

  Writes a char to the Stream message stream as a 1-byte value.
- virtual float **readFloat** () **const**

  Reads a 32 bit float from the Stream message stream.
- virtual void **writeFloat** (float value)

Writes a float to the Stream message stream as a 4 byte value.

- virtual double **readDouble** () **const**

  Reads a 64 bit double from the Stream message stream.

- virtual void **writeDouble** (double value)

  Writes a double to the Stream message stream as a 8 byte value.

- virtual short **readShort** () **const**

  Reads a 16 bit signed short from the Stream message stream.

- virtual void **writeShort** (short value)

  Writes a signed short to the Stream message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () **const**

  Reads a 16 bit unsigned short from the Stream message stream.

- virtual void **writeUnsignedShort** (unsigned short value)

  Writes a unsigned short to the Stream message stream as a 2 byte value.

- virtual int **readInt** () **const**

  Reads a 32 bit signed integer from the Stream message stream.

- virtual void **writeInt** (int value)

  Writes a signed int to the Stream message stream as a 4 byte value.

- virtual long long **readLong** () **const**

  Reads a 64 bit long from the Stream message stream.

- virtual void **writeLong** (long long value)

  Writes a long long to the Stream message stream as a 8 byte value.

- virtual std::string **readString** () **const**

  Reads an ASCII String from the Stream message stream.

- virtual void **writeString** (**const** std::string &value)

  Writes an ASCII String to the Stream message stream.

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQSTREAMMESSAGE** = 27

6.39.1 Constructor & Destructor Documentation

6.39.1.1 activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage ( )

6.39.1.2 virtual activemq::commands::ActiveMQStreamMessage::~ActiveMQStreamMessage ( ) throw () [virtual]

6.39.2 Member Function Documentation

6.39.2.1 virtual void activemq::commands::ActiveMQStreamMessage::clearBody ( ) [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

CMSException	- if an internal error occurs.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230).

6.39.2.2 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

Implements **cms::Message** (p. 1430).

6.39.2.3 virtual ActiveMQStreamMessage* activemq::commands::ActiveMQStreamMessage::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1417).

6.39.2.4 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. 1417).

6.39.2.5 virtual bool activemq::commands::ActiveMQStreamMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230).

6.39.2.6 virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1419).

6.39.2.7 virtual void activemq::commands::ActiveMQStreamMessage::onSend ( ) [virtual]

Allows derived **Message** (p. 1412) classes to perform tasks before a message is sent.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 232).

6.39.2.8 virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean ( ) const [virtual]

Reads a Boolean from the Stream 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 message stream has been reached.
MessageFormatException	- if this type conversion is invalid.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2022).

6.39.2.9 virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2022).

6.39.2.10 virtual int activemq::commands::ActiveMQStreamMessage::readBytes (std::vector< unsigned char > & *value*) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2022).

6.39.2.11 virtual int activemq::commands::ActiveMQStreamMessage::readBytes (unsigned char * *buffer,* int *length*) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2023).

6.39.2.12 virtual char activemq::commands::ActiveMQStreamMessage::readChar ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2024).

6.39.2.13 virtual double activemq::commands::ActiveMQStreamMessage::readDouble ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2024).

6.39.2.14 virtual float activemq::commands::ActiveMQStreamMessage::readFloat ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2024).

6.39.2.15 virtual int activemq::commands::ActiveMQStreamMessage::readInt ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2025).

6.39.2.16 virtual long long activemq::commands::ActiveMQStreamMessage::readLong ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2025).

6.39.2.17 virtual short activemq::commands::ActiveMQStreamMessage::readShort ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2026).

6.39.2.18 virtual std::string activemq::commands::ActiveMQStreamMessage::readString ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2026).

6.39.2.19 virtual unsigned short activemq::commands::ActiveMQStreamMessage::readUnsignedShort ( ) const [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.
MessageNotReadable-Exception	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 2026).

6.39.2.20 virtual void activemq::commands::ActiveMQStreamMessage::reset ( ) [virtual]

6.39.2.21 virtual std::string activemq::commands::ActiveMQStreamMessage::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1424).

6.39.2.22 virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean (bool *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2027).

6.39.2.23 virtual void activemq::commands::ActiveMQStreamMessage::writeByte (unsigned char *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2027).

6.39.2.24 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const std::vector< unsigned char > & *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2027).

6.39.2.25 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const unsigned char * *value*, int *offset*, int *length*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2028).

6.39.2.26 virtual void activemq::commands::ActiveMQStreamMessage::writeChar (char *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2028).

6.39.2.27 virtual void activemq::commands::ActiveMQStreamMessage::writeDouble (double *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2028).

6.39.2.28 virtual void activemq::commands::ActiveMQStreamMessage::writeFloat (float *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2029).

6.39.2.29 virtual void activemq::commands::ActiveMQStreamMessage::writeInt (int *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2029).

6.39.2.30 virtual void activemq::commands::ActiveMQStreamMessage::writeLong (long long *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2029).

6.39.2.31 virtual void activemq::commands::ActiveMQStreamMessage::writeShort (short *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2030).

6.39.2.32 virtual void activemq::commands::ActiveMQStreamMessage::writeString (const std::string & *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2030).

6.39.2.33 virtual void activemq::commands::ActiveMQStreamMessage::writeUnsignedShort (unsigned short *value*) [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.
MessageNotWriteable-Exception	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 2030).

6.39.3 Field Documentation

6.39.3.1 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**

6.40 activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 290).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQStreamMessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()

- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 290).

NOTE!: This file is 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::generated::ActiveMQStreamMessageMarshaller::-ActiveMQStreamMessageMarshaller ( ) [inline]

6.40.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQStream-MessageMarshaller::~ActiveMQStreamMessageMarshaller ( ) [inline, virtual]

6.40.3 Member Function Documentation

6.40.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal-::generated::ActiveMQStreamMessageMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.40.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.40.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.40.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.40.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.40.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.40.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQStreamMessageMarshaller.h**

6.41 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** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempDestination** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- 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.

6.41.1 Constructor & Destructor Documentation

6.41.1.1 activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ( )

6.41.1.2 activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination (const std::string & *name*)

6.41.1.3 virtual activemq::commands::ActiveMQTempDestination::∼ActiveMQTempDestination ( ) throw () [virtual]

6.41.2 Member Function Documentation

6.41.2.1 virtual ActiveMQTempDestination∗ activemq::commands::ActiveMQTempDestination::cloneDataStructure ( ) const [inline, virtual]

Clone this obbject 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::ActiveMQDestination** (p. 193).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 301), and **activemq::commands::ActiveMQTempTopic** (p. 308).

6.41.2.2 virtual void activemq::commands::ActiveMQTempDestination::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

CMSException	- If an error occurs while the resource is being closed.

Implements **cms::Closeable** (p. 633).

6.41.2.3 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

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 193).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 301), and **activemq::commands::ActiveMQTempTopic** (p. 308).

References activemq::commands::ActiveMQDestination::copyDataStructure().

6.41.2.4 virtual bool activemq::commands::ActiveMQTempDestination::equals (const DataStructure * *value*) const [inline, virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 194).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 301), and **activemq::commands::ActiveMQTempTopic** (p. 309).

References activemq::commands::ActiveMQDestination::equals().

6.41.2.5 virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 195).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 302), and **activemq::commands::ActiveMQTempTopic** (p. 309).

6.41.2.6 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 to be used to destroy this destination if closed..

6.41.2.7 virtual std::string activemq::commands::ActiveMQTempDestination::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 198).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 303), and **activemq::commands::ActiveMQTempTopic** (p. 310).

6.41.3 Field Documentation

6.41.3.1 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.

6.41.3.2 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**

6.42 activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 296).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestinationMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller-:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ∼**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.42.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (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

6.42.2 Constructor & Destructor Documentation

6.42.2.1 activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::-ActiveMQTempDestinationMarshaller ( ) [inline]

6.42.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTemp-DestinationMarshaller::∼ActiveMQTempDestinationMarshaller ( ) [inline, virtual]

6.42.3 Member Function Documentation

6.42.3.1 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestination-Marshaller::looseMarshal (OpenWireFormat ∗ *format,* commands::DataStructure ∗ *command,* decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312).

6.42.3.2 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312).

6.42.3.3 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312).

6.42.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 306), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 313).

6.42.3.5 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 306), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 313).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTempDestinationMarshaller.h**

6.43 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** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempQueue** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual **const cms::Destination** ∗ **getCMSDestination** () **const**
- virtual
  cms::Destination::DestinationType getDestinationType () **const**

  Returns the Type of Destination that this object represents.
- 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 bool **equals** (**const cms::Destination** &other) **const**

  Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getQueueName** () **const**

  Gets the name of this queue.
- virtual void **destroy** ()

  Destroy's the Temporary Destination at the Provider.

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQTEMPQUEUE** = 102

6.43.1 Constructor & Destructor Documentation

6.43.1.1 activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ( )

6.43.1.2 activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue (const std::string & *name*)

6.43.1.3 virtual activemq::commands::ActiveMQTempQueue::∼ActiveMQTempQueue ( ) throw () [virtual]

6.43.2 Member Function Documentation

6.43.2.1 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. 937).

6.43.2.2 virtual ActiveMQTempQueue* activemq::commands::ActiveMQTempQueue::cloneDataStructure ( ) const [virtual]

Clone this obbject 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::ActiveMQTempDestination** (p. 294).

6.43.2.3 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. 938).

6.43.2.4 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

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 295).

6.43.2.5 virtual void activemq::commands::ActiveMQTempQueue::destroy ( ) [virtual]

Destroy's the Temporary Destination at the Provider.

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::TemporaryQueue** (p. 2091).

6.43.2.6 virtual bool activemq::commands::ActiveMQTempQueue::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 295).

6.43.2.7 virtual bool activemq::commands::ActiveMQTempQueue::equals (const cms::Destination & *other*) const [virtual]

Compares two Destination instances to determine if they represent the same logic Destination.

Parameters

other	The other destination to compare this one to.

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 938).

6.43.2.8 virtual const cms::Destination* activemq::commands::ActiveMQTempQueue::getCMSDestination ( ) const [inline, virtual]

Returns

the **cms::Destination** (p. 936) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 195).

6.43.2.9 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

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 938).

6.43.2.10 virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 295).

6.43.2.11 virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType ( ) const [inline, virtual]

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 195).

References cms::Destination::TEMPORARY_QUEUE.

6.43.2.12 virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName ( ) const [inline, virtual]

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::TemporaryQueue** (p. 2091).

6.43.2.13 virtual std::string activemq::commands::ActiveMQTempQueue::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 296).

6.43.3 Field Documentation

6.43.3.1 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**

6.44 activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 303).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempQueueMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual ~**ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.44.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 303).

NOTE!: This file is 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::generated::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller ( ) [inline]

6.44.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::~ActiveMQTempQueueMarshaller ( ) [inline, virtual]

6.44.3 Member Function Documentation

6.44.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.44.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.44.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 297).

6.44.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298).

6.44.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298).

6.44.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299).

6.44.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTempQueueMarshaller.h**

6.45 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** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempTopic** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual **const cms::Destination** * **getCMSDestination** () **const**
- virtual
  cms::Destination::DestinationType getDestinationType () **const**

  Returns the Type of Destination that this object represents.
- 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 bool **equals** (**const cms::Destination** &other) **const**

  Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getTopicName** () **const**

  Gets the name of this topic.
- virtual void **destroy** ()

  Destroy's the Temporary Destination at the Provider.

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQTEMPTOPIC** = 103

6.45.1 Constructor & Destructor Documentation

6.45.1.1 activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ( )

6.45.1.2 activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic (const std::string & *name*)

6.45.1.3 virtual activemq::commands::ActiveMQTempTopic::~ActiveMQTempTopic ( ) throw () [virtual]

6.45.2 Member Function Documentation

6.45.2.1 virtual cms::Destination* activemq::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. 937).

6.45.2.2 virtual ActiveMQTempTopic* activemq::commands::ActiveMQTempTopic::cloneDataStructure ( ) const [virtual]

Clone this obbject 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::ActiveMQTempDestination** (p. 294).

6.45.2.3 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. 938).

6.45.2.4 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

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 295).

6.45.2.5 virtual void activemq::commands::ActiveMQTempTopic::destroy ( ) [virtual]

Destroy's the Temporary Destination at the Provider.

Exceptions

CMSException	

Implements **cms::TemporaryTopic** (p. 2092).

6.45.2.6 virtual bool activemq::commands::ActiveMQTempTopic::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 295).

6.45.2.7 virtual bool activemq::commands::ActiveMQTempTopic::equals (const cms::Destination & *other*) const [virtual]

Compares two Destination instances to determine if they represent the same logic Destination.

Parameters

other	The other destination to compare this one to.

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 938).

6.45.2.8 virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination ( ) const [inline, virtual]

Returns

the **cms::Destination** (p. 936) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 195).

6.45.2.9 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

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 938).

6.45.2.10 virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 295).

6.45.2.11 virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType ( ) const [inline, virtual]

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 195).

References cms::Destination::TEMPORARY_TOPIC.

6.45.2.12 virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName ( ) const [inline, virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::TemporaryTopic** (p. 2092).

6.45.2.13 virtual std::string activemq::commands::ActiveMQTempTopic::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 296).

6.45.3 Field Documentation

6.45.3.1 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**

6.46 activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 310).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempTopicMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 310).

NOTE!: This file is 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::generated::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]

6.46.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::~ActiveMQTempTopicMarshaller ( ) [inline, virtual]

6.46.3 Member Function Documentation

6.46.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.46.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.46.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 297).

6.46.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298).

6.46.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298).

6.46.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299).

6.46.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTempTopicMarshaller.h**

6.47 activemq::commands::ActiveMQTextMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQTextMessage.h>`

Inheritance diagram for activemq::commands::ActiveMQTextMessage:

Public Member Functions

- **ActiveMQTextMessage** ()
- virtual ~**ActiveMQTextMessage** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure*** (p. *877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTextMessage** * **cloneDataStructure** () **const**

  Clone this obbject 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. *877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure*** (p. *877) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** ()

  Clears out the body of the message.
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat)

  Called before marshaling is started to prepare the object to be marshaled.
- 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**

  Gets the message character buffer.
- virtual void **setText** (**const** char *msg)

  Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void **setText** (**const** std::string &msg)

  Sets the message contents.

Data Fields

- std::auto_ptr< std::string > **text**

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQTEXTMESSAGE** = 28

6.47.1 Constructor & Destructor Documentation

6.47.1.1 activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage ( )

6.47.1.2 virtual activemq::commands::ActiveMQTextMessage::~ActiveMQTextMessage ( ) throw () [virtual]

6.47.2 Member Function Documentation

6.47.2.1 virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal (wireformat::WireFormat * *wireFormat*) [virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

wireFormat	The wireformat object to control marshaling

Exceptions

IOException	if an I/O error occurs.

Implements **activemq::wireformat::MarshalAware** (p. 1390).

6.47.2.2 virtual void activemq::commands::ActiveMQTextMessage::clearBody ( ) [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

CMSException	- if an internal error occurs.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230).

6.47.2.3 virtual cms::TextMessage* activemq::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

Implements **cms::Message** (p. 1430).

6.47.2.4 virtual ActiveMQTextMessage* activemq::commands::ActiveMQTextMessage::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1417).

6.47.2.5 virtual void activemq::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. 1417).

6.47.2.6 virtual bool activemq::commands::ActiveMQTextMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230).

6.47.2.7 virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Message** (p. 1419).

6.47.2.8 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. 1420).

6.47.2.9 virtual std::string activemq::commands::ActiveMQTextMessage::getText ( ) const [virtual]

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::TextMessage** (p. 2093).

6.47.2.10 virtual void activemq::commands::ActiveMQTextMessage::setText (const char * *msg*) [virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

msg	The message buffer.

Exceptions

CMSException	- if an internal error occurs.
MessageNotWriteable-Exception	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 2094).

6.47.2.11 virtual void activemq::commands::ActiveMQTextMessage::setText (const std::string & *msg*) [virtual]

Sets the message contents.

Parameters

msg	The message buffer.

Exceptions

CMSException	- if an internal error occurs.
MessageNotWriteable-Exception	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 2094).

6.47.2.12 virtual std::string activemq::commands::ActiveMQTextMessage::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1424).

6.47.3 Field Documentation

6.47.3.1 const unsigned char activemq::commands::ActiveMQTextMessage::ID_ACTIVEMQTEXTMESSAGE = 28 [static]

6.47.3.2 std::auto_ptr<std::string> activemq::commands::ActiveMQTextMessage::text [mutable]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTextMessage.h**

6.48 activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessage-Marshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 318).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQ-TextMessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual ∼**ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.48.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 318).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.48.2 Constructor & Destructor Documentation

6.48.2.1 activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller ( ) [inline]

6.48.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::∼ActiveMQTextMessageMarshaller ( ) [inline, virtual]

6.48.3 Member Function Documentation

6.48.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.48.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.48.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.48.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.48.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.48.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.48.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTextMessageMarshaller.h**

6.49 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** () throw ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTopic** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual **const cms::Destination** ∗ **getCMSDestination** () **const**
- virtual
  cms::Destination::DestinationType getDestinationType () **const**

  Returns the Type of Destination that this object represents.
- 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 bool **equals** (**const cms::Destination** &other) **const**

  Compares two Destination instances to determine if they represent the same logic Destination.
- virtual std::string **getTopicName** () **const**

  Gets the name of this topic.

Static Public Attributes

- static **const** unsigned char **ID_ACTIVEMQTOPIC** = 101

6.49.1 Constructor & Destructor Documentation

6.49.1.1 activemq::commands::ActiveMQTopic::ActiveMQTopic ( )

6.49.1.2 activemq::commands::ActiveMQTopic::ActiveMQTopic (const std::string & *name*)

6.49.1.3 virtual activemq::commands::ActiveMQTopic::~ActiveMQTopic ( ) throw () [virtual]

6.49.2 Member Function Documentation

6.49.2.1 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. 937).

6.49.2.2 virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::cloneDataStructure ( ) const [virtual]

Clone this obbject 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::ActiveMQDestination** (p. 193).

6.49.2.3 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. 938).

6.49.2.4 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

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 193).

6.49.2.5 virtual bool activemq::commands::ActiveMQTopic::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 194).

6.49.2.6 virtual bool activemq::commands::ActiveMQTopic::equals (const cms::Destination & *other*) const [virtual]

Compares two Destination instances to determine if they represent the same logic Destination.

Parameters

other	The other destination to compare this one to.

Returns

true if the two destinations are the same.

Implements **cms::Destination** (p. 938).

6.49.2.7 virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination ( ) const [inline, virtual]

Returns

the **cms::Destination** (p. 936) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 195).

6.49.2.8 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

A {const} reference to a CMSProperties object.

Implements **cms::Destination** (p. 938).

6.49.2.9 virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 195).

6.49.2.10 virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType ( ) const [inline, virtual]

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

Implements **activemq::commands::ActiveMQDestination** (p. 195).

References cms::Destination::TOPIC.

6.49.2.11 virtual std::string activemq::commands::ActiveMQTopic::getTopicName ( ) const [inline, virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Topic** (p. 2142).

6.49.2.12 virtual std::string activemq::commands::ActiveMQTopic::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 198).

6.49.3 Field Documentation

6.49.3.1 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**

6.50 activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 324).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTopicMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual ~**ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.50.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 324).

NOTE!: This file is 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::generated::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller ( ) [inline]

6.50.2.2 virtual activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::~ActiveMQTopicMarshaller ( ) [inline, virtual]

6.50.3 Member Function Documentation

6.50.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.50.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.50.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

6.50.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

6.50.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201).

6.50.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202).

6.50.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ActiveMQTopicMarshaller.h**

6.51 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>`

Inheritance diagram for activemq::core::ActiveMQTransactionContext:

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. 2056) to this Transaction.*
- virtual void **removeSynchronization** (**const Pointer**< **Synchronization** > &sync)

  *Removes a **Synchronization** (p. 2056) to this Transaction.*
- virtual void **begin** ()

  Begins a new transaction if one is not currently in progress.
- virtual void **commit** ()

  Commit the current Transaction.
- virtual void **rollback** ()

  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 bool **isInLocalTransaction** () **const**

  Checks to see if there is currently an Local Transaction in progess, returns false if not, true otherwise.
- virtual bool **isInXATransaction** () **const**

  Checks to see if there is currently an XA Transaction in progess, returns false if not, true otherwise.
- virtual void **commit** (**const cms::Xid** ∗xid, bool onePhase)

  Commits a global transaction.
- virtual void **end** (**const cms::Xid** ∗xid, int flags)

  Ends the work done for a transaction branch.
- virtual void **forget** (**const cms::Xid** ∗xid)

  Informs the Resource Manager that it can forget about a specified transaction branch.
- virtual int **getTransactionTimeout** () **const**

  Gets the transaction timeout value for this XAResource.
- virtual bool **isSameRM** (**const cms::XAResource** ∗theXAResource)

  Returns true if the ResourceManager for this XAResource is the same as the Resource Manager for a supplied XAResource.
- virtual int **prepare** (**const cms::Xid** ∗xid)

  Requests the Resource manager to prepare to commit a specified transaction.
- virtual int **recover** (int flag, **cms::Xid** ∗∗recovered)

  Get a list of prepared transaction branches.
- virtual void **rollback** (**const cms::Xid** ∗xid)

  Requests the Resource Manager to rollback a specified transaction branch.
- virtual bool **setTransactionTimeout** (int seconds)

Sets the transaction timeout value for this XAResource.

- virtual void **start** (**const cms::Xid** ∗xid, int flags)

  Starts work for a specified transaction branch.

6.51.1 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.

Since

2.0

6.51.2 Constructor & Destructor Documentation

6.51.2.1 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

6.51.2.2 virtual activemq::core::ActiveMQTransactionContext::∼ActiveMQTransactionContext ( ) [virtual]

6.51.3 Member Function Documentation

6.51.3.1 virtual void activemq::core::ActiveMQTransactionContext::addSynchronization (const Pointer< Synchronization > & *sync*) [virtual]

Adds a **Synchronization** (p. 2056) to this Transaction.

Parameters

sync	- The **Synchronization** (p. 2056) instance to add.

6.51.3.2 virtual void activemq::core::ActiveMQTransactionContext::begin ( ) [virtual]

Begins a new transaction if one is not currently in progress.

Exceptions

ActiveMQException	

6.51.3.3 virtual void activemq::core::ActiveMQTransactionContext::commit ( ) [virtual]

Commit the current Transaction.

Exceptions

ActiveMQException	

6.51.3.4 virtual void activemq::core::ActiveMQTransactionContext::commit (const cms::Xid * *xid,* bool *onePhase*) [virtual]

Commits a global transaction.

Parameters

xid	the XID which identifies the global transaction.
onePhase	true if the resource manager should use a one-phase commit protocol to commit the transaction.

Exceptions

XAException	if an error occurred.

Possible errors are identified by the errorcode in the XAException and include: XA_HEURHAZ, XA_HEURCOM, XA_HEURRB, XA_HEURMIX, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO. In addition, one of the XA_RB* errors can occur if the transaction was not committed and onePhase was set to true. On completion of this method, the Resource Manager has rolled back the transaction and released resources held by the transaction.

Implements **cms::XAResource** (p. 2271).

6.51.3.5 virtual void activemq::core::ActiveMQTransactionContext::end (const cms::Xid * *xid,* int *flags*) [virtual]

Ends the work done for a transaction branch.

The Resource Manager disconnects the XA resource from the transaction branch and allows the transaction to complete.

Parameters

xid	the XID which identifies the global transaction. Should have previously been used as the parameter to a start. method.
flags	a flags integer - one of: XAResource::TMSUCCESS, XAResource::TMFAIL, or XAResource::TMSUSPEND.

TMSUCCESS means that this section of work completed successfully.

TMFAIL means that this section of work failed. The Resource Manager can mark the transaction for rollback only.

TMSUSPEND means that this section of work is suspended and not yet complete. The associated transaction context is also suspended and must be restarted with a call to **start(Xid, int)** (p. **??**) with the TMRESUME flag.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, XAER_PROTO, or XA_RB*.

Implements **cms::XAResource** (p. 2271).

6.51.3.6 virtual void activemq::core::ActiveMQTransactionContext::forget (const cms::Xid * *xid*) [virtual]

Informs the Resource Manager that it can forget about a specified transaction branch.

Parameters

xid	the XID which identifies the global transaction.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO.

Implements **cms::XAResource** (p. 2272).

6.51.3.7 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.

6.51.3.8 virtual int activemq::core::ActiveMQTransactionContext::getTransactionTimeout ( ) const [virtual]

Gets the transaction timeout value for this XAResource.

The default timeout value is the default timeout value set for the Resource Manager.

Returns

the transaction timeout value for this XAResource in seconds.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.

Implements **cms::XAResource** (p. 2272).

6.51.3.9 virtual bool activemq::core::ActiveMQTransactionContext::isInLocalTransaction ( ) const [virtual]

Checks to see if there is currently an Local Transaction in progess, returns false if not, true otherwise.

Returns

true if an Local Transaction is in progress.

6.51.3.10 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.

6.51.3.11 virtual bool activemq::core::ActiveMQTransactionContext::isInXATransaction ( ) const [virtual]

Checks to see if there is currently an XA Transaction in progess, returns false if not, true otherwise.

Returns

true if an XA Transaction is in progress.

6.51.3.12 virtual bool activemq::core::ActiveMQTransactionContext::isSameRM (const cms::XAResource * *theXAResource*) [virtual]

Returns true if the ResourceManager for this XAResource is the same as the Resource Manager for a supplied XAResource.

Parameters

theXAResource	an XAResource object

Returns

true if the Resource Manager for this XAResource is the same as the Resource Manager for `theXAResource`.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.

Implements **cms::XAResource** (p. 2272).

6.51.3.13 virtual int activemq::core::ActiveMQTransactionContext::prepare (const cms::Xid * *xid*) [virtual]

Requests the Resource manager to prepare to commit a specified transaction.

Parameters

xid	the XID which identifies the global transaction.

Returns

an integer: XA_RDONLY or XA_OK. XA_OK implies that the transaction work has been prepared normally, XA_RDONLY implies that the transaction branch is read only and has been committed. If there is a failure which requires a rollback, an XAException is raised.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XA_RB∗, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO.

Implements **cms::XAResource** (p. 2273).

6.51.3.14 virtual int activemq::core::ActiveMQTransactionContext::recover (int *flag,* cms::Xid ∗∗ *recovered*) [virtual]

Get a list of prepared transaction branches.

Typically used by a transaction manager during recovery to find transaction branches that are in prepared or heuristically completed states.

Parameters

flag	an integer. Must be one of: XAResource::TMSTARTRSCAN, XAResource::TMENDRSCAN, XAResource::TMNOFLAGS.

Returns

an array of zero or more XIDs identifying the transaction branches in the prepared or heuristically completed states.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_INVAL, and XAER_PROTO.

Implements **cms::XAResource** (p. 2273).

6.51.3.15 virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization (const Pointer< Synchronization > & *sync*) [virtual]

Removes a **Synchronization** (p. 2056) to this Transaction.

Parameters

sync	- The **Synchronization** (p. 2056) instance to add.

6.51.3.16 virtual void activemq::core::ActiveMQTransactionContext::rollback ( ) [virtual]

Rollback the current Transaction.

Exceptions

ActiveMQException	

6.51.3.17 virtual void activemq::core::ActiveMQTransactionContext::rollback (const cms::Xid ∗ *xid*) [virtual]

Requests the Resource Manager to rollback a specified transaction branch.

Parameters

xid	the XID which identifies the transaction branch.

Exceptions

XAException	if an error occurs.

Implements **cms::XAResource** (p. 2273).

6.51.3.18 virtual bool activemq::core::ActiveMQTransactionContext::setTransactionTimeout (int *seconds*) [virtual]

Sets the transaction timeout value for this XAResource.

If the value is set to 0, the default timeout value for the Resource Manager is used.

Parameters

seconds	the new Timeout value in seconds.

Returns

true if the transaction timeout value has been updated, false otherwise.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, or XAER_INVAL.

Implements **cms::XAResource** (p. 2274).

6.51.3.19 virtual void activemq::core::ActiveMQTransactionContext::start (const cms::Xid * *xid,* int *flags*) [virtual]

Starts work for a specified transaction branch.

Parameters

xid	the XID which identifies the transaction branch.
flags	an integer. Must be one of XAResource::TMNOFLAGS, XAResource::TMJOIN, or XAResource::TMRESUME.

TMJOIN implies that the start applies to joining a transaction previously passed to the Resource Manager.

TMRESUME implies that the start applies to a suspended transaction that should be restarted.

If TMNOFLAGS is specified, then if the transaction has been previously seen by the Resource Manager, an XAException is raised with the code XAER_DUPID.

Exceptions

XAException	if an error occurs. Possible error identified in the errorcode include: XA_RB*, XAER_RMERR, XAER_RMFAIL, XAER_DUPID, XAER_OUTSIDE, XAER_NOTA, XAER_INVAL, or XAER_PROTO.

Implements **cms::XAResource** (p. 2274).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQTransactionContext.h**

6.52 activemq::core::ActiveMQXAConnection Class Reference

`#include <src/main/activemq/core/ActiveMQXAConnection.h>`

Inheritance diagram for activemq::core::ActiveMQXAConnection:

Public Member Functions

- **ActiveMQXAConnection** (**const Pointer**< **transport::Transport** > &transport, **const Pointer**< **decaf::util::Properties** > &properties)
- virtual ~**ActiveMQXAConnection** () throw ()
- virtual **cms::XASession** * **createXASession** ()

  Creates an XASession object.
- virtual **cms::Session** * **createSession** (**cms::Session::AcknowledgeMode** ackMode)

  *Creates a new **Session** (p. 1830) to work for this **Connection** (p. 725) using the specified acknowledgment mode.*

  Parameters

ackMode	*the Acknowledgment Mode to use.*

  Exceptions

CMSException (p. 640)	

6.52.1 Constructor & Destructor Documentation

6.52.1.1 activemq::core::ActiveMQXAConnection::ActiveMQXAConnection (const Pointer< transport::Transport > & *transport,* const Pointer< decaf::util::Properties > & *properties*)

6.52.1.2 virtual activemq::core::ActiveMQXAConnection::~ActiveMQXAConnection ( ) throw () [virtual]

6.52.2 Member Function Documentation

6.52.2.1 virtual cms::Session* activemq::core::ActiveMQXAConnection::createSession (cms::Session::AcknowledgeMode *ackMode*) [virtual]

Creates a new **Session** (p. 1830) to work for this **Connection** (p. 725) using the specified acknowledgment mode.

Parameters

ackMode	the Acknowledgment Mode to use.

Exceptions

CMSException *(p. 640)*	

Reimplemented from **activemq::core::ActiveMQConnection** (p. 152).

6.52.2.2 virtual cms::XASession* activemq::core::ActiveMQXAConnection::createXASession ( ) [virtual]

Creates an XASession object.

Returns

a newly created XASession instance, caller owns the pointer.

Exceptions

CMSException	If the XAConnection object fails to create the XASession instance due to an internal error.

Implements **cms::XAConnection** (p. 2262).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQXAConnection.h**

6.53 activemq::core::ActiveMQXAConnectionFactory Class Reference

`#include <src/main/activemq/core/ActiveMQXAConnectionFactory.h>`

Inheritance diagram for activemq::core::ActiveMQXAConnectionFactory:

Public Member Functions

- **ActiveMQXAConnectionFactory** ()
- **ActiveMQXAConnectionFactory** (**const** std::string &uri, **const** std::string &username="", **const** std::string &password="")

  Constructor.
- **ActiveMQXAConnectionFactory** (**const decaf::net::URI** &uri, **const** std::string &username="", **const** std::string &password="")

  Constructor.
- virtual ~**ActiveMQXAConnectionFactory** () throw ()
- virtual **cms::XAConnection** * **createXAConnection** ()

  Creates an XAConnection with the default user name and password.
- virtual **cms::XAConnection** * **createXAConnection** (**const** std::string &userName, **const** std::string &password)

  Creates an XA connection with the specified user name and password.

Protected Member Functions

- virtual **ActiveMQConnection** * **createActiveMQConnection** (**const Pointer**< **transport::Transport** > &transport, **const Pointer**< **decaf::util::Properties** > &properties)

  *Create a new **ActiveMQConnection** (p. 145) instnace using the provided Transport and Properties.*

6.53.1 Constructor & Destructor Documentation

6.53.1.1 activemq::core::ActiveMQXAConnectionFactory::ActiveMQXAConnectionFactory ( )

6.53.1.2 activemq::core::ActiveMQXAConnectionFactory::ActiveMQXAConnectionFactory (const std::string & *uri,* const std::string & *username* = " ", const std::string & *password* = " ")

Constructor.

Parameters

uri	the URI of the Broker we are connecting to.
username	to authenticate with, defaults to ""
password	to authenticate with, defaults to ""

6.53.1.3 activemq::core::ActiveMQXAConnectionFactory::ActiveMQXAConnectionFactory (const decaf::net::URI & *uri,* const std::string & *username* = " ", const std::string & *password* = " ")

Constructor.

Parameters

uri	the URI of the Broker we are connecting to.
username	to authenticate with, defaults to ""
password	to authenticate with, defaults to ""

6.53.1.4 virtual activemq::core::ActiveMQXAConnectionFactory::~ActiveMQXAConnectionFactory ( ) throw () [virtual]

6.53.2 Member Function Documentation

6.53.2.1 virtual ActiveMQConnection* activemq::core::ActiveMQXAConnectionFactory::createActiveMQConnection (const Pointer< transport::Transport > & *transport,* const Pointer< decaf::util::Properties > & *properties*) [protected, virtual]

Create a new **ActiveMQConnection** (p. 145) instnace using the provided Transport and Properties.

Subclasses can override this to control the actual type of **ActiveMQConnection** (p. 145) that is created.

Parameters

transport	The Transport that the Connection should use to communicate with the Broker.
properties	The Properties that are assigned to the new Connection instance.

Returns

a new **ActiveMQConnection** (p. 145) pointer instance.

Reimplemented from **activemq::core::ActiveMQConnectionFactory** (p. 168).

6.53.2.2 virtual cms::XAConnection* activemq::core::ActiveMQXAConnectionFactory::createXAConnection ( ) [virtual]

Creates an XAConnection with the default user name and password.

The connection is created in stopped mode just as the standard Connection object is created from the ConnectionFactory. No messages will be delivered until the Connection.start method is explicitly called.

Returns

a new XAConnectionFactory instance, the caller owns the returned pointer.

Exceptions

CMSException	if an internal error occurs while creating the Connection.
CMSSecurityException	if the client authentication fails because the user name or password are invalid.

Implements **cms::XAConnectionFactory** (p. 2264).

6.53.2.3 virtual cms::XAConnection* activemq::core::ActiveMQXAConnectionFactory::createXAConnection (const std::string & *userName,* const std::string & *password*) [virtual]

Creates an XA connection with the specified user name and password.

The connection is created in stopped mode just as the standard ConnectionFactory creates a new Connection. No messages will be delivered until the Connection.start method is explicitly called.

Returns

a new XAConnectionFactory instance, the caller owns the returned pointer.

Exceptions

CMSException	if an internal error occurs while creating the Connection.
CMSSecurityException	if the client authentication fails because the user name or password are invalid.

Implements **cms::XAConnectionFactory** (p. 2264).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQXAConnectionFactory.h**

6.54 activemq::core::ActiveMQXASession Class Reference

`#include <src/main/activemq/core/ActiveMQXASession.h>`

Inheritance diagram for activemq::core::ActiveMQXASession:

Public Member Functions

- **ActiveMQXASession** (**ActiveMQConnection** *__connection__, **const Pointer**< **commands::SessionId** > &sessionId, **const decaf::util::Properties** &properties)
- virtual ~**ActiveMQXASession** () throw ()
- virtual bool **isTransacted** () **const**

  Gets if the Sessions is a Transacted Session.
- virtual bool **isAutoAcknowledge** () **const**
- virtual void **doStartTransaction** ()

  Starts if not already start a Transaction for this Session.
- virtual void **commit** ()

  Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()

Rolls back all messages done in this transaction and releases any locks currently held.

- virtual **cms::XAResource** * **getXAResource** () **const**

  Returns the XA resource associated with this Session to the caller.

6.54.1 Constructor & Destructor Documentation

6.54.1.1 activemq::core::ActiveMQXASession::ActiveMQXASession (ActiveMQConnection * *connection,* const Pointer< commands::SessionId > & *sessionId,* const decaf::util::Properties & *properties*)

6.54.1.2 virtual activemq::core::ActiveMQXASession::~ActiveMQXASession ( ) throw () [virtual]

6.54.2 Member Function Documentation

6.54.2.1 virtual void activemq::core::ActiveMQXASession::commit ( ) [virtual]

Commits 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.

Reimplemented from **activemq::core::ActiveMQSession** (p. 262).

6.54.2.2 virtual void activemq::core::ActiveMQXASession::doStartTransaction ( ) [virtual]

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.

Reimplemented from **activemq::core::ActiveMQSession** (p. 269).

6.54.2.3 virtual cms::XAResource* activemq::core::ActiveMQXASession::getXAResource ( ) const [virtual]

Returns the XA resource associated with this Session to the caller.

The client can use the provided XA resource to interact with the XA Transaction Manager in use in the client application.

Returns

an XAResouce instance to the caller, the caller does not own this pointer and should not delete it.

Implements **cms::XASession** (p. 2276).

6.54.2.4 virtual bool activemq::core::ActiveMQXASession::isAutoAcknowledge ( ) const [virtual]

Reimplemented from **activemq::core::ActiveMQSession** (p. 271).

6.54.2.5 virtual bool activemq::core::ActiveMQXASession::isTransacted ( ) const [virtual]

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

CMSException	- If an internal error occurs.

Reimplemented from **activemq::core::ActiveMQSession** (p. 271).

6.54.2.6 virtual void activemq::core::ActiveMQXASession::rollback ( ) [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.

Reimplemented from **activemq::core::ActiveMQSession** (p. 273).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQXASession.h**

6.55 decaf::util::zip::Adler32 Class Reference

Clas that can be used to compute an Adler-32 **Checksum** (p. 628) for a data stream.

`#include <src/main/decaf/util/zip/Adler32.h>`

Inheritance diagram for decaf::util::zip::Adler32:

Public Member Functions

- **Adler32** ()
- virtual ∼**Adler32** ()
- virtual long long **getValue** () **const**
- virtual void **reset** ()

  Reset the checksum to its initial value.
- virtual void **update** (**const** std::vector< unsigned char > &buffer)

  Updates the current checksum with the specified vector of bytes.
- virtual void **update** (**const** std::vector< unsigned char > &buffer, int offset, int length)

  Updates the current checksum with the specified array of bytes.
- virtual void **update** (**const** unsigned char *buffer, int size, int offset, int length)

  Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)

  Updates the current checksum with the specified byte value.

6.55.1 Detailed Description

Clas that can be used to compute an Adler-32 **Checksum** (p. 628) for a data stream.

The Alder-32 checksum trades reliability for speed over the CRC-32 algorithm.

Since

1.0

6.55.2 Constructor & Destructor Documentation

6.55.2.1 decaf::util::zip::Adler32::Adler32 ( )

6.55.2.2 virtual decaf::util::zip::Adler32::~Adler32 ( ) [virtual]

6.55.3 Member Function Documentation

6.55.3.1 virtual long long decaf::util::zip::Adler32::getValue ( ) const [virtual]

Returns

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 629).

6.55.3.2 virtual void decaf::util::zip::Adler32::reset ( ) [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 629).

6.55.3.3 virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & *buffer*) [virtual]

Updates the current checksum with the specified vector of bytes.

Parameters

buffer	The buffer to read the updated bytes from.

Implements **decaf::util::zip::Checksum** (p. 629).

6.55.3.4 virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & *buffer,* int *offset,* int *length*) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer	The buffer to read the updated bytes from.
offset	The position in the buffer to start reading.
length	The amount of data to read from the byte buffer.

Exceptions

IndexOutOfBounds-Exception	if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 629).

6.55.3.5 virtual void decaf::util::zip::Adler32::update (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer	The buffer to read the updated bytes from.
size	The size of the passed buffer.
offset	The position in the buffer to start reading.
length	The amount of data to read from the byte buffer.

Exceptions

NullPointerException	if the passed buffer is NULL.
IndexOutOfBounds-Exception	if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 629).

6.55.3.6 virtual void decaf::util::zip::Adler32::update (int *byte*) [virtual]

Updates the current checksum with the specified byte value.

Parameters

byte	The byte value to update the current **Checksum** (p. 628) with (0..255).

Implements **decaf::util::zip::Checksum** (p. 630).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Adler32.h**

6.56 decaf::lang::Appendable Class Reference

An object to which char sequences and values can be appended.

`#include <src/main/decaf/lang/Appendable.h>`

Inheritance diagram for decaf::lang::Appendable:

Public Member Functions

- virtual ~**Appendable** ()
- virtual **Appendable** & **append** (char value)=0

  *Appends the specified character to this **Appendable** (p. 342).*
- virtual **Appendable** & **append** (**const CharSequence** *csq)=0

  *Appends the specified character sequence to this **Appendable** (p. 342).*
- virtual **Appendable** & **append** (**const CharSequence** *csq, int start, int end)=0

  *Appends a subsequence of the specified character sequence to this **Appendable** (p. 342).*

6.56.1 Detailed Description

An object to which char sequences and values can be appended.

The **Appendable** (p. 342) interface must be implemented by any class whose instances are intended to receive formatted output from a Formatter.

TODO The characters to be appended should be valid Unicode characters as described in Unicode **Character** (p. 593) Representation. Note that supplementary characters may be composed of multiple 16-bit char values.

Appendables are not necessarily safe for multithreaded access. **Thread** (p. 2094) safety is the responsibility of classes that extend and implement this interface.

Since this interface may be implemented by existing classes with different styles of error handling there is no guarantee that errors will be propagated to the invoker.

Since

1.0

6.56.2 Constructor & Destructor Documentation

6.56.2.1 virtual decaf::lang::Appendable::~Appendable ( ) [inline, virtual]

6.56.3 Member Function Documentation

6.56.3.1 virtual Appendable& decaf::lang::Appendable::append (char *value*) [pure virtual]

Appends the specified character to this **Appendable** (p. 342).

Parameters

value	The character to append.

Returns

a Reference to this **Appendable** (p. 342)

Exceptions

Exception *(p. 990)*	if an error occurs.

Implemented in **decaf::io::Writer** (p. 2256), and **decaf::nio::CharBuffer** (p. 612).

6.56.3.2 virtual Appendable& decaf::lang::Appendable::append (const CharSequence * *csq*) [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 342).

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. 342).

Exceptions

Exception *(p. 990)*	if an error occurs.

Implemented in **decaf::io::Writer** (p. 2256), and **decaf::nio::CharBuffer** (p. 612).

6.56.3.3 virtual Appendable& decaf::lang::Appendable::append (const CharSequence * *csq,* int *start,* int *end*) [pure virtual]

Appends a subsequence of the specified character sequence to this **Appendable** (p. 342).

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. 342)

Exceptions

Exception *(p. 990)*	if an error occurs.
IndexOutOfBounds-Exception	start is greater than end, or end is greater than csq.length()

Implemented in **decaf::nio::CharBuffer** (p. 613), and **decaf::io::Writer** (p. 2257).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Appendable.h**

6.57 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.

6.57.1 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.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 decaf::internal::AprPool::AprPool ( )

6.57.2.2 virtual decaf::internal::AprPool::~AprPool ( ) [virtual]

6.57.3 Member Function Documentation

6.57.3.1 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.

6.57.3.2 apr_pool_t* decaf::internal::AprPool::getAprPool ( ) const

Gets the internal APR Pool.

Returns

the internal APR pool

6.57.3.3 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**

6.58 decaf::util::ArrayList< E > Class Template Reference

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

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

Public Member Functions

- **ArrayList** ()
- **ArrayList** (**const Collection**< E > &collection)
- **ArrayList** (**const ArrayList**< E > &arrayList)
- **ArrayList** (int initialCapacity)

- virtual ~**ArrayList** ()
- **ArrayList**< E > & **operator=** (**const ArrayList**< E > &list)
- **ArrayList**< E > & **operator=** (**const Collection**< E > &collection)
- void **ensureCapacity** (int minimumCapacity)

  *Increases the capacity of this **ArrayList** (p. 345) instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.*
- void **trimToSize** ()

  *Trims the internal array to match the current size of the **ArrayList** (p. 345).*
- virtual void **clear** ()

  Removes all of the elements from this collection (optional operation).
- virtual bool **isEmpty** () **const**

  Returns true if this collection contains no elements.
- virtual int **size** () **const**

  Returns the number of elements in this collection.
- virtual E **set** (int index, **const** E &element)

  Replaces the element at the specified position in this list with the specified element.
- virtual E **get** (int index) **const**

  Gets the element contained at position passed.
- virtual bool **add** (**const** E &value)

  Returns true if this collection changed as a result of the call.
- virtual void **add** (int index, **const** E &element)

  Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (**const Collection**< E > &collection)

  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	*The **Collection** (p. 660) whose elements are added to this one.*

  Returns

  true if this collection changed as a result of the call

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
IllegalArgumentException	*if some property of an element prevents it from being added to this collection*
IllegalStateException	*if an element cannot be added at this time due to insertion restrictions.*

  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).
- virtual bool **addAll** (int index, **const Collection**< E > &collection)

  Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual bool **remove** (**const** E &value)

  Removes a single instance of the specified element from the collection.
  More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this **Collection** (p. 660).*

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

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.

- virtual E **removeAt** (int index)

  Removes the element at the specified position in this list.

- virtual bool **contains** (**const** E &value) **const**

  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 (value == NULL ? e == NULL : value == e).

Parameters

value	*The value to check for presence in the collection.*

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

- virtual int **indexOf** (**const** E &value) **const**

  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 int **lastIndexOf** (**const** E &value) **const**

  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 std::vector< E > **toArray** () **const**

  *Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).*

- virtual std::string **toString** () **const**

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

6.58.1 Constructor & Destructor Documentation

6.58.1.1 template<typename E> decaf::util::ArrayList< E >::ArrayList ( ) [inline]

References decaf::util::ArrayList< E >::ensureCapacity().

6.58.1.2 template<typename E> decaf::util::ArrayList< E >::ArrayList (const Collection< E > & *collection*) [inline]

References decaf::lang::Iterable< E >::iterator(), and decaf::util::Collection< E >::size().

6.58.1.3 template<typename E> decaf::util::ArrayList< E >::ArrayList (const ArrayList< E > & *arrayList*) [inline]

References decaf::util::AbstractList< E >::iterator(), and decaf::util::ArrayList< E >::size().

6.58.1.4 template<typename E> decaf::util::ArrayList< E >::ArrayList (int *initialCapacity*) [inline]

6.58.1.5 template<typename E> virtual decaf::util::ArrayList< E >::~ArrayList ( ) [inline, virtual]

References DECAF_CATCHALL_NOTHROW.

6.58.2 Member Function Documentation

6.58.2.1 template<typename E> virtual bool decaf::util::ArrayList< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Reimplemented from **decaf::util::AbstractList< E >** (p. 98).

6.58.2.2 template<typename E> virtual void decaf::util::ArrayList< E >::add (int *index*, const E & *element*) [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	The index at which the specified element is to be inserted.
element	The element to be inserted in this **List** (p. 1286).

Exceptions

IndexOutOfBounds-Exception	if the index is greater than size of the **List** (p. 1286).
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List< E >** (p. 1287).

6.58.2.3 template<typename E> virtual bool decaf::util::ArrayList< E >::addAll (const Collection< E > & *collection*) [inline, 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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

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).

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).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 87).

References decaf::util::Collection< E >::size(), and decaf::util::Collection< E >::toArray().

Referenced by decaf::util::ArrayList< E >::operator=().

6.58.2.4 template<typename E> virtual bool decaf::util::ArrayList< E >::addAll (int *index*, const Collection< E > & *source*) [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. 660) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Reimplemented from **decaf::util::AbstractList< E >** (p. 99).

References decaf::util::Collection< E >::size(), and decaf::util::Collection< E >::toArray().

6.58.2.5 template<typename E> virtual void decaf::util::ArrayList< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractList< E >** (p. 100).

Referenced by decaf::util::ArrayList< E >::operator=().

6.58.2.6 template<typename E> virtual bool decaf::util::ArrayList< E >::contains (const E & *value*) const [inline, 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 (value == NULL ? e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 88).

References decaf::util::ArrayList< E >::indexOf().

6.58.2.7 template<typename E> void decaf::util::ArrayList< E >::ensureCapacity (int *minimumCapacity*) [inline]

Increases the capacity of this **ArrayList** (p. 345) instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

If the capacity is already greater than or equal to the minimum capacity argument then the array is left unchanged.

Parameters

minimum-Capacity	The desired minimum capacity for this **ArrayList** (p. 345).

References decaf::lang::System::arraycopy().

Referenced by decaf::util::ArrayList< E >::ArrayList().

6.58.2.8 template<typename E> virtual E decaf::util::ArrayList< E >::get (int *index*) const [inline, virtual]

Gets the element contained at position passed.

Parameters

index	The position to get.

Returns

value at index specified.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.

Implements **decaf::util::List< E >** (p. 1289).

6.58.2.9 template<typename E> virtual int decaf::util::ArrayList< E >::indexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Reimplemented from **decaf::util::AbstractList< E >** (p. 100).

Referenced by decaf::util::ArrayList< E >::contains(), and decaf::util::ArrayList< E >::remove().

6.58.2.10 template<typename E> virtual bool decaf::util::ArrayList< E >::isEmpty ( ) const [inline, virtual]

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 354) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

6.58.2.11 template<typename E> virtual int decaf::util::ArrayList< E >::lastIndexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Reimplemented from **decaf::util::AbstractList< E >** (p. 102).

6.58.2.12 template<typename E> ArrayList<E>& decaf::util::ArrayList< E >::operator= (const ArrayList< E > & *list*) [inline]

References decaf::util::ArrayList< E >::addAll(), and decaf::util::ArrayList< E >::clear().

6.58.2.13 template<typename E> ArrayList<E>& decaf::util::ArrayList< E >::operator= (const Collection< E > & *collection*) [inline]

References decaf::util::ArrayList< E >::addAll(), and decaf::util::ArrayList< E >::clear().

6.58.2.14 template<typename E> virtual bool decaf::util::ArrayList< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 92).

References decaf::util::ArrayList< E >::indexOf(), and decaf::util::ArrayList< E >::removeAt().

6.58.2.15 template<typename E> virtual E decaf::util::ArrayList< E >::removeAt (int *index*) [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

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

Reimplemented from **decaf::util::AbstractList< E >** (p. 104).

References decaf::lang::System::arraycopy().

Referenced by decaf::util::ArrayList< E >::remove().

6.58.2.16 template<typename E> virtual E decaf::util::ArrayList< E >::set (int *index*, const E & *element*) [inline, virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters

index	The index of the element to replace.
element	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List< E >** (p. 1294).

6.58.2.17 template<typename E> virtual int decaf::util::ArrayList< 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. 669).

Referenced by decaf::util::ArrayList< E >::ArrayList().

6.58.2.18 template<typename E> virtual std::vector<E> decaf::util::ArrayList< E >::toArray ( ) const [inline, virtual]

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).

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. 660)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 94).

6.58.2.19 template<typename E> virtual std::string decaf::util::ArrayList< E >::toString ( ) const [inline, virtual]

References decaf::lang::Integer::toString().

6.58.2.20 template<typename E> void decaf::util::ArrayList< E >::trimToSize ( ) [inline]

Trims the internal array to match the current size of the **ArrayList** (p. 345).

This compacts the internal array to save storage space used by this **ArrayList** (p. 345).

References decaf::lang::System::arraycopy().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**ArrayList.h**

6.59 decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator Class Reference

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

Inheritance diagram for decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator:

Public Member Functions

- **ArrayListIterator** (**const ArrayPointer**< E > &array, int index)
- virtual ~**ArrayListIterator** ()
- virtual E **next** ()

  Returns the next element in the iteration.
- virtual bool **hasNext** () **const**

  Returns true if the iteration has more elements.
- virtual void **remove** ()

  Removes from the underlying collection the last element returned by the iterator (optional operation).
- virtual void **add** (**const** E &e **DECAF_UNUSED**)
- virtual void **set** (**const** E &e **DECAF_UNUSED**)

- virtual bool **hasPrevious** () **const**

  Returns true if this list iterator has more elements when traversing the list in the reverse direction.

- virtual E **previous** ()

  Returns the previous element in the list.

- virtual int **nextIndex** () **const**

  Returns the index of the element that would be returned by a subsequent call to next.

- virtual int **previousIndex** () **const**

  Returns the index of the element that would be returned by a subsequent call to previous.

template<typename E> class decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator

6.59.1 Constructor & Destructor Documentation

6.59.1.1 template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::ArrayListIterator (const ArrayPointer< E > & *array,* int *index*) [inline]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::length().

6.59.1.2 template<typename E> virtual decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::~ArrayListIterator ( ) [inline, virtual]

6.59.2 Member Function Documentation

6.59.2.1 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::add (const E &e *DECAF_UNUSED*) [inline, virtual]

6.59.2.2 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::hasNext ( ) const [inline, virtual]

Returns true if the iteration has more elements.

Returns false if the next call to next would result in an **NoSuchElementException** (p. 1537) to be thrown.

Returns

true if there are more elements available for iteration.

Implements **decaf::util::Iterator< E >** (p. 1209).

6.59.2.3 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::hasPrevious ( ) const [inline, 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.

Implements **decaf::util::ListIterator< E >** (p. 1296).

6.59.2.4 template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::next ( ) [inline, virtual]

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p. 356) method returns false will return each element in the underlying collection exactly once.

Returns

the next element in the iteration of elements.

Exceptions

NoSuchElementException *(p. 1537)*	if the iteration has no more elements.

Implements **decaf::util::Iterator< E >** (p. 1209).

6.59.2.5 template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::nextIndex ( ) const [inline, 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.

Implements **decaf::util::ListIterator< E >** (p. 1296).

6.59.2.6 template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::previous ( ) [inline, 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 *(p. 1537)*	if the iteration has no previous element.

Implements **decaf::util::ListIterator< E >** (p. 1296).

6.59.2.7 template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::previousIndex ( ) const [inline, 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.

Implements **decaf::util::ListIterator< E >** (p. 1297).

6.59.2.8 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::remove ( ) [inline, 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

UnsupportedOperation-Exception	if the remove operation is not supported by this **Iterator** (p. 1209).
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.

Implements **decaf::util::Iterator< E >** (p. 1210).

6.59.2.9 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::set (const E &e *DECAF_UNUSED*) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CopyOnWriteArrayList.h**

6.60 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 1614) that is a template on a Type and is **Thread** (p. 2094) Safe if the default Reference Counter is used.

`#include <src/main/decaf/lang/ArrayPointer.h>`

Data Structures

- struct **ArrayData**

Public Types

- typedef T * **PointerType**
- typedef T & **ReferenceType**
- typedef **const** T & **ConstReferenceType**
- typedef REFCOUNTER **CounterType**

Public Member Functions

- **ArrayPointer** ()

  Default Constructor.

- **ArrayPointer** (int size)

  *Create a new **ArrayPointer** (p. 358) instance and allocates an internal array that is sized using the passed in size value.*
- **ArrayPointer** (int size, **const** T &fillWith)

  *Create a new **ArrayPointer** (p. 358) instance and allocates an internal array that is sized using the passed in size value.*
- **ArrayPointer** (**const PointerType** value, int size)

  *Explicit Constructor, creates an **ArrayPointer** (p. 358) that contains value with a single reference.*
- **ArrayPointer** (**const ArrayPointer** &value) throw ()

  Copy constructor.
- virtual ~**ArrayPointer** () throw ()
- void **reset** (T *value, int size=0)

  *Resets the **ArrayPointer** (p. 358) to hold the new value.*
- T * **release** ()

  *Releases the **Pointer** (p. 1614) held and resets the internal pointer value to Null.*
- **PointerType get** () **const**

  *Gets the real array pointer that is contained within this **Pointer** (p. 1614).*
- int **length** () **const**

  Returns the current size of the contained array or zero if the array is NULL.
- void **swap** (**ArrayPointer** &value) throw ()

  ***Exception** (p. 990) Safe Swap Function.*
- **ArrayPointer clone** () **const**

  *Creates a new **ArrayPointer** (p. 358) instance that is a clone of the value contained in this **ArrayPointer** (p. 358).*
- **ArrayPointer** & **operator=** (**const ArrayPointer** &right) throw ()

  *Assigns the value of right to this **Pointer** (p. 1614) and increments the reference Count.*
- template<typename T1 , typename R1 >
  ArrayPointer & **operator=** (**const ArrayPointer**< T1, R1 > &right) throw ()
- **ReferenceType operator[]** (int index)

  Dereference Operator, returns a reference to the Contained value.
- **ConstReferenceType operator[]** (int index) **const**
- bool **operator!** () **const**
- template<typename T1 , typename R1 >
  bool **operator==** (**const ArrayPointer**< T1, R1 > &right) **const**
- template<typename T1 , typename R1 >
  bool **operator!=** (**const ArrayPointer**< T1, R1 > &right) **const**

Friends

- bool **operator==** (**const ArrayPointer** &left, **const** T *right)
- bool **operator==** (**const** T *left, **const ArrayPointer** &right)
- bool **operator!=** (**const ArrayPointer** &left, **const** T *right)
- bool **operator!=** (**const** T *left, **const ArrayPointer** &right)

6.60.1 Detailed Description

template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter>class decaf::lang::ArrayPointer< T, REFCOUNTER >

Decaf's implementation of a Smart **Pointer** (p. 1614) that is a template on a Type and is **Thread** (p. 2094) Safe if the default Reference Counter is used.

This **Pointer** (p. 1614) 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. 1614) 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. 1614) in a STL container that requires it, **Pointer** (p. 1614) provides an implementation of std::less.

Since

1.0

6.60.2 Member Typedef Documentation

6.60.2.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef const T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ConstReferenceType

6.60.2.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER decaf::lang::ArrayPointer< T, REFCOUNTER >::CounterType

6.60.2.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T* decaf::lang::ArrayPointer< T, REFCOUNTER >::PointerType

6.60.2.4 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ReferenceType

6.60.3 Constructor & Destructor Documentation

6.60.3.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer ( ) [inline]

Default Constructor.

Initialized the contained array pointer to NULL, using the subscript operator results in an exception unless reset to contain a real value.

Referenced by decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::clone(), and decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::reset().

6.60.3.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (int *size*) [inline]

Create a new **ArrayPointer** (p. 358) instance and allocates an internal array that is sized using the passed in size value.

Parameters

size	The size of the array to allocate for this **ArrayPointer** (p. 358) instance.

6.60.3.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (int *size,* const T & *fillWith*) [inline]

Create a new **ArrayPointer** (p. 358) instance and allocates an internal array that is sized using the passed in size value.

The array elements are initialized with the given value.

Parameters

size	The size of the array to allocate for this **ArrayPointer** (p. 358) instance.
fillWith	The value to initialize each element of the newly allocated array with.

6.60.3.4 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (const PointerType *value,* int *size*) [inline, explicit]

Explicit Constructor, creates an **ArrayPointer** (p. 358) that contains value with a single reference.

This object now has ownership until a call to release.

Parameters

value	The pointer to the instance of the array we are taking ownership of.
size	The size of the array this object is taking ownership of.

6.60.3.5 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (const ArrayPointer< T, REFCOUNTER > & *value*) throw () [inline]

Copy constructor.

Copies the value contained in the **ArrayPointer** (p. 358) to the new instance and increments the reference counter.

6.60.3.6 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> virtual decaf::lang::ArrayPointer< T, REFCOUNTER >::~ArrayPointer ( ) throw () [inline, virtual]

6.60.4 Member Function Documentation

6.60.4.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer decaf::lang::ArrayPointer< T, REFCOUNTER >::clone ( ) const [inline]

Creates a new **ArrayPointer** (p. 358) instance that is a clone of the value contained in this **ArrayPointer** (p. 358).

Returns

an **ArrayPointer** (p. 358) that contains a copy of the data in this **ArrayPointer** (p. 358).

6.60.4.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> PointerType decaf::lang::ArrayPointer< T, REFCOUNTER >::get ( ) const [inline]

Gets the real array pointer that is contained within this **Pointer** (p. 1614).

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. 1614). Use at your own risk.

Returns

the contained pointer.

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent(), decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::clone(), decaf::lang::ArrayPointerComparator< T, R >::compare(), decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::operator!=(), decaf::lang::operator!=(), decaf::lang::ArrayPointerComparator< T, R >::operator()(), std::less< decaf::lang::ArrayPointer< T > >::operator()(), decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::operator==(), decaf::lang::operator==(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set().

6.60.4.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> int decaf::lang::ArrayPointer< T, REFCOUNTER >::length ( ) const [inline]

Returns the current size of the contained array or zero if the array is NULL.

Returns

the size of the array or zero if the array is NULL

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator::ArrayListIterator(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::contains(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::get(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::indexOf(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::lastIndexOf(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::toArray().

6.60.4.4 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator! ( ) const [inline]

6.60.4.5 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 , typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator!= (const ArrayPointer< T1, R1 > & *right*) const [inline]

6.60.4.6 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer& decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= (const ArrayPointer< T, REFCOUNTER > & *right*) throw () [inline]

Assigns the value of right to this **Pointer** (p. 1614) and increments the reference Count.

Parameters

right	- **Pointer** (p. 1614) on the right hand side of an operator= call to this.

6.60.4.7 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 , typename R1 > ArrayPointer& decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= (const ArrayPointer< T1, R1 > & *right*) throw () [inline]

6.60.4.8 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 , typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator== (const ArrayPointer< T1, R1 > & *right*) const [inline]

6.60.4.9 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] (int *index*) [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

6.60.4.10 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ConstReferenceType decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] (int *index*) const [inline]

6.60.4.11 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> T* decaf::lang::ArrayPointer< T, REFCOUNTER >::release ( ) [inline]

Releases the **Pointer** (p. 1614) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 1614) 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. 1614) object, the pointer is no longer owned by this **Pointer** (p. 1614) and won't be freed when this **Pointer** (p. 1614) goes out of scope.

Referenced by decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::ArrayPointer().

6.60.4.12 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::ArrayPointer< T, REFCOUNTER >::reset (T * *value,* int *size* = 0) [inline]

Resets the **ArrayPointer** (p. 358) 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. 1614) to a NULL pointer.

Parameters

value	The new array pointer value to contain.
size	The size of the new array value this object now contains.

6.60.4.13 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::ArrayPointer< T, REFCOUNTER >::swap (ArrayPointer< T, REFCOUNTER > & *value*) throw () [inline]

Exception (p. 990) Safe Swap Function.

Parameters

value	- the value to swap with this.

Referenced by decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::operator=(), and decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::swap().

6.60.5 Friends And Related Function Documentation

6.60.5.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const ArrayPointer< T, REFCOUNTER > & *left,* const T * *right*) [friend]

6.60.5.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const T * *left,* const ArrayPointer< T, REFCOUNTER > & *right*) [friend]

6.60.5.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== (const ArrayPointer< T, REFCOUNTER > & *left,* const T * *right*) [friend]

6.60.5.4 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== (const T * *left,* const ArrayPointer< T, REFCOUNTER > & *right*) [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

6.61 decaf::lang::ArrayPointerComparator< 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 value of the actual pointer to the array being contained in this **ArrayPointer** (p. 358).

`#include <src/main/decaf/lang/ArrayPointer.h>`

Inheritance diagram for decaf::lang::ArrayPointerComparator< T, R >:

Public Member Functions

- virtual ~**ArrayPointerComparator** ()
- virtual bool **operator()** (**const ArrayPointer**< T, R > &left, **const ArrayPointer**< T, R > &right) **const**
- virtual int **compare** (**const ArrayPointer**< T, R > &left, **const ArrayPointer**< T, R > &right) **const**

6.61.1 Detailed Description

template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter>class decaf::lang::ArrayPointerComparator< 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 value of the actual pointer to the array being contained in this **ArrayPointer** (p. 358).

This allows for a basic ordering to be acheived in Decaf containers.

Custom implementations are possible where an array of some type has a logical natural ordering such as array of integers where the sum of all ints in the array is used.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual decaf::lang::ArrayPointerComparator< T, R >::~ArrayPointerComparator ( ) [inline, virtual]

6.61.3 Member Function Documentation

6.61.3.1 template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual int decaf::lang::ArrayPointerComparator< T, R >::compare (const ArrayPointer< T, R > & *left,* const ArrayPointer< T, R > & *right*) const [inline, virtual]

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

6.61.3.2 template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool decaf::lang::ArrayPointerComparator< T, R >::operator() (const ArrayPointer< T, R > & *left,* const ArrayPointer< T, R > & *right*) const [inline, virtual]

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

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

6.62 decaf::util::Arrays Class Reference

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

Public Member Functions

- virtual ~**Arrays** ()

Static Public Member Functions

- template<typename E >
  static void **fill** (E *array, int size, **const** E &value)

  Fills an array with the specified element.
- template<typename E >
  static void **fill** (E *array, int size, int start, int end, **const** E &value)

  Fills an array with the specified element within the range given.

6.62.1 Constructor & Destructor Documentation

6.62.1.1 virtual decaf::util::Arrays::~Arrays ( ) [virtual]

6.62.2 Member Function Documentation

6.62.2.1 template<typename E > static void decaf::util::Arrays::fill (E * *array,* int *size,* const E & *value*) [inline, static]

Fills an array with the specified element.

Parameters

array	The Array to fill.
size	The actual size of the array passed.
value	The value to fill the array with.

Exceptions

NullPointerException	if array is Null.
IllegalArgumentException	if the size parameter is negative, or the start index is greater than the end index.

Referenced by decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::ArrayPointer().

6.62.2.2 template<typename E > static void decaf::util::Arrays::fill (E * *array*, int *size*, int *start*, int *end*, const E & *value*) [inline, static]

Fills an array with the specified element within the range given.

Parameters

array	The Array to fill.
size	The actual size of the array passed.
start	The index to start the fill from (inclusive).
end	The index to fill to (exclusive).
value	The value to fill the array with.

Exceptions

NullPointerException	if array is Null.
IllegalArgumentException	if the size parameter is negative, or the start index is greater than the end index.
IndexOutOfBounds-Exception	if the start index is negative or the end index is greater than the size parameter.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Arrays.h**

6.63 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. 366) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)

  *Creates a new **AtomicBoolean** (p. 366) with the initial value.*
- virtual ~**AtomicBoolean** ()
- bool **get** () **const**

  *Gets the current value of this **AtomicBoolean** (p. 366).*
- 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.

6.63.1 Detailed Description

A boolean value that may be updated atomically.

An **AtomicBoolean** (p. 366) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ( )

Creates a new **AtomicBoolean** (p. 366) whose initial value is false.

6.63.2.2 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean (bool *initialValue*)

Creates a new **AtomicBoolean** (p. 366) with the initial value.

Parameters

initialValue	- The initial value of this boolean.

6.63.2.3 virtual decaf::util::concurrent::atomic::AtomicBoolean::~AtomicBoolean ( ) [inline, virtual]

6.63.3 Member Function Documentation

6.63.3.1 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.

6.63.3.2 bool decaf::util::concurrent::atomic::AtomicBoolean::get ( ) const [inline]

Gets the current value of this **AtomicBoolean** (p. 366).

Returns

the currently set value.

6.63.3.3 bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet (bool *newValue*)

Atomically sets to the given value and returns the previous value.

Parameters

newValue	- the new value

Returns

the previous value

6.63.3.4 void decaf::util::concurrent::atomic::AtomicBoolean::set (bool *newValue*) [inline]

Unconditionally sets to the given value.

Parameters

newValue	- the new value

6.63.3.5 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**

6.64 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. *368) with an initial value of 0.*
- **AtomicInteger** (int initialValue)

  *Create a new **AtomicInteger** (*p. *368) 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.

6.64.1 Detailed Description

An int value that may be updated atomically.

An **AtomicInteger** (p. 368) 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.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ( )

Create a new **AtomicInteger** (p. 368) with an initial value of 0.

6.64.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger (int *initialValue*)

Create a new **AtomicInteger** (p. 368) with the given initial value.

Parameters

initialValue	- The initial value of this object.

6.64.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~AtomicInteger ( ) [inline, virtual]

6.64.3 Member Function Documentation

6.64.3.1 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.

6.64.3.2 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.

6.64.3.3 int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ( )

Atomically decrements by one the current value.

Returns

the updated value.

Referenced by decaf::util::concurrent::atomic::AtomicRefCounter::release().

6.64.3.4 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. 1544).

6.64.3.5 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. 1544).

6.64.3.6 int decaf::util::concurrent::atomic::AtomicInteger::get ( ) const [inline]

Gets the current value.

Returns

the current value.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo(), decaf::util::concurrent::LinkedBlockingQueue< E >::offer(), decaf::util::concurrent::LinkedBlockingQueue< E >::peek(), decaf::util::concurrent::LinkedBlockingQueue< E >::poll(), decaf::util::concurrent::LinkedBlockingQueue< E >::put(), decaf::util::concurrent::LinkedBlockingQueue< E >::remainingCapacity(), decaf::util::concurrent::LinkedBlockingQueue< E >::size(), decaf::util::concurrent::LinkedBlockingQueue< E >::take(), decaf::util::concurrent::LinkedBlockingQueue< E >::toArray(), and decaf::util::concurrent::LinkedBlockingQueue< E >::toString().

6.64.3.7 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.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo().

6.64.3.8 int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ( )

Atomically decrements by one the current value.

Returns

the previous value.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::poll(), and decaf::util::concurrent::LinkedBlockingQueue< E >::take().

6.64.3.9 int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ( )

Atomically increments by one the current value.

Returns

the previous value.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::offer(), and decaf::util::concurrent::LinkedBlockingQueue< E >::put().

6.64.3.10 int decaf::util::concurrent::atomic::AtomicInteger::getAndSet (int *newValue*)

Atomically sets to the given value and returns the old value.

Parameters

newValue	- the new value.

Returns

the previous value.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::clear().

6.64.3.11 int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ( )

Atomically increments by one the current value.

Returns

the updated value.

Referenced by decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter().

6.64.3.12 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. 1544).

6.64.3.13 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. 1545).

6.64.3.14 void decaf::util::concurrent::atomic::AtomicInteger::set (int *newValue*) [inline]

Sets to the given value.

Parameters

newValue	- the new value

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::clear().

6.64.3.15 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**

6.65 decaf::util::concurrent::atomic::AtomicRefCounter Class Reference

`#include <src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h>`

Inherited by **decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >**, **decaf::lang::ArrayPointer< E >**, **decaf::lang::ArrayPointer< ServiceListener * >**, **decaf::lang::ArrayPointer< unsigned char >**, **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< ByteArrayAdapter >**, **decaf::lang::Pointer< CloseTransportsTask >**, **decaf::lang::Pointer< cms::Destination >**, **decaf::lang::Pointer< commands::ActiveMQDestination >**, **decaf::lang::Pointer< commands::ConsumerId >**, **decaf::lang::Pointer< commands::ConsumerInfo >**, **decaf::lang::Pointer< commands::Message >**, **decaf::lang::Pointer< commands::ProducerInfo >**, **decaf::lang::Pointer< commands::SessionInfo >**, **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< Exception >**, **decaf::lang::Pointer< LogManagerInternals >**, **decaf::lang::Pointer< Message >**, **decaf::lang::Pointer< MessageAck >**, **decaf::lang::Pointer< MessageDispatchChannel >**, **decaf::lang::Pointer< MessageId >**, **decaf::lang::Pointer< ProducerId >**, **decaf::lang::Pointer< ProducerInfo >**, **decaf::lang::Pointer< Properties >**, **decaf::lang::Pointer< QueueNode< E > >**, **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< TransactionState >**, **decaf::lang::Pointer< 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)
- virtual ~**AtomicRefCounter** ()

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.

6.65.1 Constructor & Destructor Documentation

6.65.1.1 decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter ( ) [inline]

6.65.1.2 decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter (const AtomicRefCounter & *other*) [inline]

References decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet().

6.65.1.3 virtual decaf::util::concurrent::atomic::AtomicRefCounter::~AtomicRefCounter ( ) [inline, virtual]

6.65.2 Member Function Documentation

6.65.2.1 bool decaf::util::concurrent::atomic::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::ArrayPointer< ServiceListener * >** (p. 363), **decaf::lang::ArrayPointer< E >** (p. 363), **decaf::lang::ArrayPointer< unsigned char >** (p. 363), **decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >** (p. 363), **decaf::lang::Pointer< MessageAck >** (p. 1619), **decaf::lang::Pointer< BooleanExpression >** (p. 1619), **decaf::lang::Pointer< commands::ConsumerId >** (p. 1619), **decaf::lang::Pointer< BrokerError >** (p. 1619), **decaf::lang::Pointer< Transport >** (p. 1619), **decaf::lang::Pointer< wireformat::WireFormat >** (p. 1619), **decaf::lang::Pointer< MessageDispatchChannel >** (p. 1619), **decaf::lang::Pointer< commands::WireFormatInfo >** (p. 1619), **decaf::lang::Pointer< CloseTransportsTask >** (p. 1619), **decaf::lang::Pointer< CompositeTaskRunner >** (p. 1619), **decaf::lang::Pointer< ActiveMQTransactionContext >** (p. 1619), **decaf::lang::Pointer< commands::ProducerInfo >** (p. 1619), **decaf::lang::Pointer< Comparator< E > >** (p. 1619), **decaf::lang::Pointer< BrokerId >** (p. 1619), **decaf::lang::Pointer< LogManagerInternals >** (p. 1619), **decaf::lang::Pointer< commands::SessionInfo >** (p. 1619), **decaf::lang::Pointer< Message >** (p. 1619), **decaf::lang::Pointer< URI >** (p. 1619), **decaf::lang::Pointer< DataStructure >** (p. 1619), **decaf::lang::Pointer< activemq::threads::TaskRunner >** (p. 1619), **decaf::lang::Pointer< commands::ActiveMQDestination >** (p. 1619), **decaf::lang::Pointer< ConsumerInfo >** (p. 1619), **decaf::lang::Pointer< ConnectionId >** (p. 1619), **decaf::lang::Pointer< decaf::lang::Runnable >** (p. 1619), **decaf::lang::Pointer< Properties >** (p. 1619), **decaf::lang::Pointer< BackupTransportPool >** (p. 1619), **decaf::lang::Pointer< ProducerInfo >** (p. 1619), **decaf::lang::Pointer< decaf::lang::Thread >** (p. 1619), **decaf::lang::Pointer< MessageId >** (p. 1619), **decaf::lang::Pointer< Response >** (p. 1619), **decaf::lang::Pointer< QueueNode< E > >** (p. 1619), **decaf::lang::Pointer< SessionId >** (p. 1619), **decaf::lang::Pointer< cms::Destination >** (p. 1619), **decaf::lang::Pointer< TransportListener >** (p. 1619), **decaf::lang::Pointer< ActiveMQDestination >** (p. 1619), **decaf::lang::Pointer< ProducerId >** (p. 1619), **decaf::lang::Pointer< ResponseBuilder >** (p. 1619), **decaf::lang::Pointer< SessionInfo >** (p. 1619), **decaf::lang::Pointer< commands::Message >** (p. 1619), **decaf::lang::Pointer< Tracked >** (p. 1619), **decaf::lang::Pointer< ConnectionInfo >** (p. 1619), **decaf::lang::Pointer< core::ActiveMQAckHandler >** (p. 1619), **decaf::lang::Pointer< Exception >** (p. 1619), **decaf::lang::Pointer< TransactionState >** (p. 1619), **decaf::lang::Pointer< commands::ConsumerInfo >** (p. 1619), **decaf::lang::Pointer< ConsumerId >** (p. 1619), **decaf::lang::Pointer< URIPool >** (p. 1619), **decaf::lang::Pointer< ByteArrayAdapter >** (p. 1619), and **decaf::lang::Pointer< TransactionId >** (p. 1619).

References decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet().

6.65.2.2 void decaf::util::concurrent::atomic::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/util/concurrent/atomic/**AtomicRefCounter.h**

6.66 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.

6.66.1 Detailed Description

template<typename T>class decaf::util::concurrent::atomic::AtomicReference< T >

An Pointer reference that may be updated atomically.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 template<typename T > decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference ( ) [inline]

6.66.2.2 template<typename T > decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference (T * *value*) [inline]

6.66.2.3 template<typename T > virtual decaf::util::concurrent::atomic::AtomicReference< T >::~AtomicReference ( ) [inline, virtual]

6.66.3 Member Function Documentation

6.66.3.1 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.

6.66.3.2 template<typename T > T* decaf::util::concurrent::atomic::AtomicReference< T >::get ( ) const [inline]

Gets the Current Value.

Returns

the current value of this Reference.

6.66.3.3 template<typename T > T* decaf::util::concurrent::atomic::AtomicReference< T >::getAndSet (T * *newValue*) [inline]

Atomically sets to the given value and returns the old value.

Parameters

newValue-	the new value

Returns

the previous value.

6.66.3.4 template<typename T > void decaf::util::concurrent::atomic::AtomicReference< T >::set (T * *newValue*) [inline]

Sets the Current value of this Reference.

Parameters

newValue	The new Value of this Reference.

6.66.3.5 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**

6.67 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. 2161).*
- **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. 2161) been shutdown and no longer usable.*
- void **setClosed** (bool value)

  *Sets the closed flag on this **Transport** (p. 2161).*

6.67.1 Constructor & Destructor Documentation

6.67.1.1 activemq::transport::failover::BackupTransport::BackupTransport (BackupTransportPool * *failover*)

6.67.1.2 virtual activemq::transport::failover::BackupTransport::~BackupTransport ( ) [virtual]

6.67.2 Member Function Documentation

6.67.2.1 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.

6.67.2.2 decaf::net::URI activemq::transport::failover::BackupTransport::getUri ( ) const [inline]

Gets the URI assigned to this Backup.

Returns

the assigned URI

6.67.2.3 bool activemq::transport::failover::BackupTransport::isClosed ( ) const [inline]

Has the **Transport** (p. 2161) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2161)

6.67.2.4 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. 377) closes its internal **Transport** (p. 2161) 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. 2177).

6.67.2.5 void activemq::transport::failover::BackupTransport::setClosed (bool *value*) [inline]

Sets the closed flag on this **Transport** (p. 2161).

Parameters

value	- true for closed.

6.67.2.6 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.

6.67.2.7 void activemq::transport::failover::BackupTransport::setUri (const decaf::net::URI & *uri*) [inline]

Sets the URI assigned to this **Transport** (p. 2161).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransport.h**

6.68 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. 2161) 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. 2161) 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. 2161) Pool is enabled.*

Friends

- class **BackupTransport**

6.68.1 Constructor & Destructor Documentation

6.68.1.1 activemq::transport::failover::BackupTransportPool::BackupTransportPool (const Pointer< CompositeTaskRunner > & *taskRunner,* const Pointer< CloseTransportsTask > & *closeTask,* const Pointer< URIPool > & *uriPool*)

6.68.1.2 activemq::transport::failover::BackupTransportPool::BackupTransportPool (int *backupPoolSize,* const Pointer< CompositeTaskRunner > & *taskRunner,* const Pointer< CloseTransportsTask > & *closeTask,* const Pointer< URIPool > & *uriPool*)

6.68.1.3 virtual activemq::transport::failover::BackupTransportPool::~BackupTransportPool ( ) [virtual]

6.68.2 Member Function Documentation

6.68.2.1 Pointer<BackupTransport> activemq::transport::failover::BackupTransportPool::getBackup ( )

Get a Connected **Transport** (p. 2161) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

Returns

Pointer to a Connected **Transport** (p. 2161) or NULL

6.68.2.2 int activemq::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.

6.68.2.3 bool activemq::transport::failover::BackupTransportPool::isEnabled ( ) const [inline]

Gets if the backup **Transport** (p. 2161) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

Returns

true if enable.

6.68.2.4 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. 692).

6.68.2.5 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. 2073).

6.68.2.6 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.

6.68.2.7 void activemq::transport::failover::BackupTransportPool::setEnabled (bool *value*)

Sets if this Backup **Transport** (p. 2161) 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.

6.68.3 Friends And Related Function Documentation

6.68.3.1 friend class BackupTransport [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransportPool.h**

6.69 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. 671) Id of this **Message** (p. 1412).*
- virtual int **getCommandId** () **const**

  *Gets the **Command** (p. 671) Id of this **Message** (p. 1412).*
- virtual void **setResponseRequired** (**const** bool required)

  *Set if this **Message** (p. 1412) requires a **Response** (p. 1781).*
- virtual bool **isResponseRequired** () **const**

  *Is a **Response** (p. 1781) required for this **Command** (p. 671).*
- 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionControl** () **const**
- 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**

6.69.1 Constructor & Destructor Documentation

6.69.1.1 activemq::commands::BaseCommand::BaseCommand ( ) [inline]

6.69.1.2 virtual activemq::commands::BaseCommand::~BaseCommand ( ) [inline, virtual]

6.69.2 Member Function Documentation

6.69.2.1 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. 879).

Reimplemented in **activemq::commands::Message** (p. 1417), **activemq::commands::ConsumerInfo** (p. 785), **activemq::commands::BrokerError** (p. 434), **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::BrokerInfo** (p. 445), **activemq::commands::ConnectionInfo** (p. 753), **activemq::commands::MessageAck** (p. 1449), **activemq::commands::ConsumerControl** (p. 771), **activemq::commands::ProducerInfo** (p. 1699), **activemq::commands::ConnectionControl** (p. 730), **activemq::commands::DestinationInfo** (p. 940), **activemq::commands::MessagePull** (p. 1504), **activemq::commands::SessionInfo** (p. 1850), **activemq::commands::MessageDispatch** (p. 1460), **activemq::commands::MessageDispatchNotification** (p. 1470), **activemq::commands::TransactionInfo** (p. 2150), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::RemoveSubscriptionInfo** (p. 1765), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ProducerAck** (p. 1684), **activemq::commands::RemoveInfo** (p. 1759), **activemq::commands::DataArrayResponse** (p. 829), **activemq::commands::DataResponse** (p. 864), **activemq::commands::ExceptionResponse** (p. 997), **activemq::commands::ReplayCommand** (p. 1771), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::Response** (p. 1782), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::FlushCommand** (p. 1069), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::ShutdownInfo** (p. 1884), **activemq::commands::ActiveMQBlobMessage** (p. 123), **activemq::commands::WireFormatInfo** (p. 2242), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 224).

References getCommandId(), and isResponseRequired().

6.69.2.2 virtual bool activemq::commands::BaseCommand::equals (const DataStructure * *value*) const [inline, virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

Reimplemented in **activemq::commands::Message** (p. 1417), **activemq::commands::ConsumerInfo** (p. 785), **activemq::commands::ActiveMQBytesMessage** (p. 132), **activemq::commands::BrokerInfo** (p. 445), **activemq::commands::ConnectionInfo** (p. 753), **activemq::commands::MessageAck** (p. 1450), **activemq::commands::ConsumerControl** (p. 771), **activemq::commands::ProducerInfo** (p. 1699), **activemq::commands::ConnectionControl** (p. 730), **activemq::commands::DestinationInfo** (p. 941),

activemq::commands::MessagePull (p. 1504), **activemq::commands::SessionInfo** (p. 1850), **activemq::commands::MessageDispatch** (p. 1460), **activemq::commands::MessageDispatchNotification** (p. 1470), **activemq::commands::TransactionInfo** (p. 2150), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::RemoveSubscriptionInfo** (p. 1765), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ProducerAck** (p. 1684), **activemq::commands::RemoveInfo** (p. 1759), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 864), **activemq::commands::ExceptionResponse** (p. 998), **activemq::commands::ReplayCommand** (p. 1772), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::Response** (p. 1783), **activemq::commands::FlushCommand** (p. 1069), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230), **activemq::commands::ActiveMQBlobMessage** (p. 124), **activemq::commands::WireFormatInfo** (p. 2242), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 224).

References activemq::commands::BaseDataStructure::equals().

6.69.2.3 virtual int activemq::commands::BaseCommand::getCommandId ( ) const [inline, virtual]

Gets the **Command** (p. 671) Id of this **Message** (p. 1412).

Returns

Command (p. 671) Id

Implements **activemq::commands::Command** (p. 671).

Referenced by copyDataStructure().

6.69.2.4 virtual bool activemq::commands::BaseCommand::isBrokerInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::BrokerInfo** (p. 446).

6.69.2.5 virtual bool activemq::commands::BaseCommand::isConnectionControl ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::ConnectionControl** (p. 730).

6.69.2.6 virtual bool activemq::commands::BaseCommand::isConnectionInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::ConnectionInfo** (p. 754).

6.69.2.7 virtual bool activemq::commands::BaseCommand::isConsumerInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::ConsumerInfo** (p. 786).

6.69.2.8 virtual bool activemq::commands::BaseCommand::isKeepAliveInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::KeepAliveInfo** (p. 1235).

6.69.2.9 virtual bool activemq::commands::BaseCommand::isMessage ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::Message** (p. 1421).

6.69.2.10 virtual bool activemq::commands::BaseCommand::isMessageAck ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::MessageAck** (p. 1451).

6.69.2.11 virtual bool activemq::commands::BaseCommand::isMessageDispatch ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::MessageDispatch** (p. 1461).

6.69.2.12 virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 672).

Reimplemented in **activemq::commands::MessageDispatchNotification** (p. 1471).

6.69.2.13 virtual bool activemq::commands::BaseCommand::isProducerAck ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::ProducerAck** (p. 1685).

6.69.2.14 virtual bool activemq::commands::BaseCommand::isProducerInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::ProducerInfo** (p. 1700).

6.69.2.15 virtual bool activemq::commands::BaseCommand::isRemoveInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::RemoveInfo** (p. 1760).

6.69.2.16 virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::RemoveSubscriptionInfo** (p. 1766).

6.69.2.17 virtual bool activemq::commands::BaseCommand::isResponse ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::Response** (p. 1783).

6.69.2.18 virtual bool activemq::commands::BaseCommand::isResponseRequired ( ) const [inline, virtual]

Is a **Response** (p. 1781) required for this **Command** (p. 671).

Returns

true if a response is required.

Implements **activemq::commands::Command** (p. 673).

Referenced by copyDataStructure().

6.69.2.19 virtual bool activemq::commands::BaseCommand::isShutdownInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 1885).

6.69.2.20 virtual bool activemq::commands::BaseCommand::isTransactionInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::TransactionInfo** (p. 2151).

6.69.2.21 virtual bool activemq::commands::BaseCommand::isWireFormatInfo ( ) const [inline, virtual]

Implements **activemq::commands::Command** (p. 673).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 2245).

6.69.2.22 virtual void activemq::commands::BaseCommand::setCommandId (int *id*) [inline, virtual]

Sets the **Command** (p. 671) Id of this **Message** (p. 1412).

Parameters

id	**Command** (p. 671) Id

Implements **activemq::commands::Command** (p. 674).

6.69.2.23 virtual void activemq::commands::BaseCommand::setResponseRequired (const bool *required*) [inline, virtual]

Set if this **Message** (p. 1412) requires a **Response** (p. 1781).

Parameters

required	true if response is required

Implements **activemq::commands::Command** (p. 674).

6.69.2.24 virtual std::string activemq::commands::BaseCommand::toString ( ) const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements **activemq::commands::Command** (p. 674).

Reimplemented in **activemq::commands::Message** (p. 1424), **activemq::commands::ConsumerInfo** (p. 788), **activemq::commands::ActiveMQBytesMessage** (p. 138), **activemq::commands::BrokerInfo** (p. 447), **activemq::commands::ConnectionInfo** (p. 755), **activemq::commands::MessageAck** (p. 1451), **activemq::commands::ConsumerControl** (p. 772), **activemq::commands::ProducerInfo** (p. 1700), **activemq::commands::ConnectionControl** (p. 731), **activemq::commands::DestinationInfo** (p. 942), **activemq::commands::MessagePull** (p. 1505), **activemq::commands::SessionInfo** (p. 1851), **activemq::commands::MessageDispatch** (p. 1461), **activemq::commands::MessageDispatchNotification** (p. 1472), **activemq::commands::TransactionInfo** (p. 2151), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::RemoveSubscriptionInfo** (p. 1766), **activemq::commands::ActiveMQStreamMessage** (p. 286), **activemq::commands::ProducerAck** (p. 1685), **activemq::commands::RemoveInfo** (p. 1760), **activemq::commands::ActiveMQMapMessage** (p. 219), **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 865), **activemq::commands::ExceptionResponse** (p. 998), **activemq::commands::ReplayCommand** (p. 1772), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::Response** (p. 1783), **activemq::commands::FlushCommand** (p. 1070), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::ActiveMQBlobMessage** (p. 125), **activemq::commands::WireFormatInfo** (p. 2247), **activemq::commands::ActiveMQTextMessage** (p. 317), **activemq::commands::ActiveMQObjectMessage** (p. 235), and **activemq::commands::ActiveMQMessage** (p. 225).

References activemq::commands::BaseDataStructure::toString().

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BaseCommand.h**

6.70 activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 386).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/BaseCommandMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual ~**BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.70.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (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

6.70.2 Constructor & Destructor Documentation

6.70.2.1 activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::BaseCommandMarshaller ( ) [inline]

6.70.2.2 virtual activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::~BaseCommandMarshaller ( ) [inline, virtual]

6.70.3 Member Function Documentation

6.70.3.1 virtual void activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 221), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 449), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 734), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 740), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 757), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 774), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 790), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 796), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 944), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1178), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1237), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1453), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1467), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1474), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1507), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1687), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1762), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1769), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1774), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1887), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2153), and **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487).

6.70.3.2 virtual void activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319), **activemq::wireformat::**

::openwire::marshal::generated::BrokerInfoMarshaller (p. 450), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 734), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 740), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 757), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 774), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 791), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 797), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 944), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1238), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1467), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1474), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1508), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1687), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1762), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1769), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1775), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1887), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2153), and **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.70.3.3 virtual int activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 450), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 734), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 740), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 758), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 775), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 791), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 797), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire-**

::marshal::generated::DataResponseMarshaller (p. 867), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1238), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1468), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1475), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1508), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1688), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1763), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1769), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1775), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1888), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2154), and **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488).

6.70.3.4 virtual void activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 145), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 450), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 735), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 741), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 758), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 775), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 791), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 797), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1238), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller**

(p. 1454), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1468), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1475), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1508), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1688), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1704), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1763), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1770), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1775), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1854), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1888), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2154), and **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

6.70.3.5 virtual void activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 145), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 238), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 451), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 735), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 741), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 759), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 776), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 792), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 798), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 868), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1180), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1239), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1455), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1468), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1475), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1509), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1688), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1704), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1764), **activemq::wireformat::openwire::marshal::generated::**

::RemoveSubscriptionInfoMarshaller (p. 1770), **activemq::wireformat::openwire::marshal::generated::-ReplayCommandMarshaller** (p. 1776), **activemq::wireformat::openwire::marshal::generated::Response-Marshaller** (p. 1792), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1854), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1889), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2154), and **activemq-::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**BaseCommandMarshaller.h**

6.71 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

`#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStream-Marshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller:

Public Member Functions

- virtual ~**BaseDataStreamMarshaller** ()
- virtual int **tightMarshal1** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataOutputStream** *ds AMQCPP_UNUSED, **utils::Boolean-Stream** *bs AMQCPP_UNUSED)

  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::Boolean-Stream** *bs AMQCPP_UNUSED)

  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)

  Tight Marshal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataInputStream** *dis AMQCPP_UNUSED)

  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)

  Tight Unmarshal the cached object.
- virtual int **tightMarshalCachedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **utils::BooleanStream** *bs)

  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)

  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)

  Loosely marshals the passed DataStructure based object to the passed stream returning nothing.
- virtual **commands::DataStructure** * **looseUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn)

  Loose Unmarshal the cached object.
- virtual int **tightMarshalNestedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **utils::BooleanStream** *bs)

  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)

  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)

  Tight Unmarshal the nested object.
- virtual **commands::DataStructure** * **looseUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn)

  Loose Unmarshal the nested object.
- virtual void **looseMarshalNestedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut)

  Loose marshall the nested object.
- virtual std::string **tightUnmarshalString** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Performs Tight Unmarshaling of String Objects.
- virtual int **tightMarshalString1** (**const** std::string &value, **utils::BooleanStream** *bs)

  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)

  Tight Marshals the passed string to the streams passed.
- virtual void **looseMarshalString** (**const** std::string value, **decaf::io::DataOutputStream** *dataOut)

  Loose Marshal the String to the DataOuputStream passed.
- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream** *dataIn)

  Loose Un-Marshal the String to the DataOuputStream passed.
- virtual int **tightMarshalLong1** (**OpenWireFormat** *wireFormat, long long value, **utils::BooleanStream** *bs)

  Tightly marshal the long long to the BooleanStream passed.
- virtual void **tightMarshalLong2** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tightly marshal the long long to the Streams passed.
- virtual long long **tightUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight marshal the long long type.

- virtual void **looseMarshalLong** (**OpenWireFormat** ∗wireFormat, long long value, **decaf::io::DataOutputStream** ∗dataOut)

  Tightly marshal the long long to the BooleanStream passed.

- virtual long long **looseUnmarshalLong** (**OpenWireFormat** ∗wireFormat, **decaf::io::DataInputStream** ∗dataIn)

  Loose marshal the long long type.

- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Unmarshal an array of char.

- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** ∗dataIn)

  Loose Unmarshal an array of char.

- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs, int size)

  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)

  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)

  Tight Unarshall the Error object.

- virtual int **tightMarshalBrokerError1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗data, **utils::BooleanStream** ∗bs)

  Tight Marshal the Error object.

- virtual void **tightMarshalBrokerError2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗data, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marshal the Error object.

- virtual **commands::DataStructure** ∗ **looseUnmarshalBrokerError** (**OpenWireFormat** ∗wireFormat, **decaf::io::DataInputStream** ∗dataIn)

  Loose Unarshal the Error object.

- virtual void **looseMarshalBrokerError** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗data, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marshal the Error object.

- template<typename T >
  int **tightMarshalObjectArray1** (**OpenWireFormat** ∗wireFormat, std::vector< T > objects, **utils::BooleanStream** ∗bs)

  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)

  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 > objects, **decaf::io::DataOutputStream** ∗dataOut)

  Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- virtual std::string **readAsciiString** (**decaf::io::DataInputStream** ∗dataIn)

  Given an DataInputStream read a know ASCII formatted string from the input and return that string.

6.71.1 Detailed Description

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

Since

2.0

6.71.2 Constructor & Destructor Documentation

6.71.2.1 virtual activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamMarshaller ( ) [inline, virtual]

6.71.3 Member Function Documentation

6.71.3.1 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED) [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.

6.71.3.2 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerError (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) [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.

6.71.3.3 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedObject (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) [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.

6.71.3.4 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*) [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.

6.71.3.5 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedObject (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*) [protected, virtual]

Loose marshall 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.

6.71.3.6 template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray (OpenWireFormat * *wireFormat*, std::vector< T > *objects*, decaf::io::DataOutputStream * *dataOut*) [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, AMQ_CATCHALL_THROW, decaf::io::DataOutputStream::writeBoolean(), and decaf::io::DataOutputStream::writeShort().

6.71.3.7 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString (const std::string *value*, decaf::io::DataOutputStream * *dataOut*) [protected, virtual]

Loose Marshal the String to the DataOuputStream passed.

Parameters

value	- string to marshal
dataOut	- stream to write marshaled form to

Exceptions

IOException	if an error occurs.

6.71.3.8 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED) [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.

6.71.3.9 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBrokerError (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) [protected, virtual]

Loose Unarshal 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.

6.71.3.10 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalByteArray (decaf::io::DataInputStream * *dataIn*) [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.

6.71.3.11 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCachedObject (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) [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.

6.71.3.12 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConstByteArray (decaf::io::DataInputStream * *dataIn*, int *size*) [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 unmarshaled vector of chars.

Exceptions

IOException	if an error occurs.

6.71.3.13 virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) [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.

6.71.3.14 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNestedObject (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) [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.

6.71.3.15 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString (decaf::io::DataInputStream * *dataIn*) [protected, virtual]

Loose Un-Marshal the String to the DataOuputStream passed.

Parameters

dataIn	- stream to read marshaled form from

Returns

the unmarshaled string

Exceptions

IOException	if an error occurs.

6.71.3.16 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString (decaf::io::DataInputStream * *dataIn*) [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

6.71.3.17 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) [inline, virtual]

Tight Marhsal 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.

6.71.3.18 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) [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.

6.71.3.19 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, utils::BooleanStream * *bs*) [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.

6.71.3.20 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [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.

6.71.3.21 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedObject1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, utils::BooleanStream * *bs*) [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.

6.71.3.22 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedObject2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [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.

6.71.3.23 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1 (OpenWireFormat * *wireFormat*, long long *value*, utils::BooleanStream * *bs*) [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.

6.71.3.24 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2 (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [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.

6.71.3.25 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, utils::BooleanStream * *bs*) [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.

6.71.3.26 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [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.

6.71.3.27 template<typename T > int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1 (OpenWireFormat * *wireFormat*, std::vector< T > *objects*, utils::BooleanStream * *bs*) [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, AMQ_CATCHALL_THROW, and activemq::wireformat::openwire::utils::BooleanStream::writeBoolean().

6.71.3.28 template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2 (OpenWireFormat * *wireFormat*, std::vector< T > *objects*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [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, AMQ_CATCHALL_THROW, activemq::wireformat::openwire::utils::BooleanStream::readBoolean(), and decaf::io::DataOutputStream::writeShort().

6.71.3.29 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1 (const std::string & *value*, utils::BooleanStream * *bs*) [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.

6.71.3.30 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2 (const std::string & *value*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) [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.

6.71.3.31 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) [inline, virtual]

Tight Un-Marshal 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 Un-Marshal from.

Exceptions

IOException	if an error occurs.

6.71.3.32 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBrokerError (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) [protected, virtual]

Tight Unarshall 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.

6.71.3.33 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByteArray (decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) [protected, virtual]

Tight Unmarshal an array of char.

Parameters

dataIn	- the DataInputStream to Un-Marshal from
bs	- boolean stream to unmarshal from.

Returns

the unmarshaled vector of chars.

Exceptions

IOException	if an error occurs.

6.71.3.34 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCachedObject (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) [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.

6.71.3.35 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConstByteArray (decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*, int *size*) [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
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.

6.71.3.36 virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) [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.

6.71.3.37 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNestedObject (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) [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.

6.71.3.38 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString (decaf::io::DataInputStream * *dataIn,* utils::BooleanStream * *bs*) [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.

6.71.3.39 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

6.71.3.40 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().

6.71.3.41 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

6.71.3.42 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

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**BaseDataStreamMarshaller.h**

6.72 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 the class implementing this interface is really wanting to be told about marshaling.
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)
- virtual void **afterMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)
- virtual void **beforeUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)
- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)

- virtual void **setMarshaledForm** (**wireformat::WireFormat** ∗wireFormat AMQCPP_UNUSED, **const** std::vector< char > &data AMQCPP_UNUSED)
- virtual std::vector< unsigned char > **getMarshaledForm** (**wireformat::WireFormat** ∗wireFormat AMQCPP_UNUSED)
- virtual void **copyDataStructure** (**const DataStructure** ∗src AMQCPP_UNUSED)
- virtual std::string **toString** () **const**

  *Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value AMQCPP_UNUSED) **const**

6.72.1 Constructor & Destructor Documentation

6.72.1.1 virtual activemq::commands::BaseDataStructure::~BaseDataStructure ( ) [inline, virtual]

6.72.2 Member Function Documentation

6.72.2.1 virtual void activemq::commands::BaseDataStructure::afterMarshal (wireformat::WireFormat ∗wireFormat *AMQCPP_UNUSED*) [inline, virtual]

6.72.2.2 virtual void activemq::commands::BaseDataStructure::afterUnmarshal (wireformat::WireFormat ∗wireFormat *AMQCPP_UNUSED*) [inline, virtual]

Reimplemented in **activemq::commands::WireFormatInfo** (p. 2242), and **activemq::commands::Message** (p. 1416).

6.72.2.3 virtual void activemq::commands::BaseDataStructure::beforeMarshal (wireformat::WireFormat ∗wireFormat *AMQCPP_UNUSED*) [inline, virtual]

Reimplemented in **activemq::commands::WireFormatInfo** (p. 2242), and **activemq::commands::Message** (p. 1417).

6.72.2.4 virtual void activemq::commands::BaseDataStructure::beforeUnmarshal (wireformat::WireFormat ∗wireFormat *AMQCPP_UNUSED*) [inline, virtual]

6.72.2.5 virtual void activemq::commands::BaseDataStructure::copyDataStructure (const DataStructure ∗src *AMQCPP_UNUSED*) [inline, virtual]

Reimplemented in **activemq::commands::BooleanExpression** (p. 427).

6.72.2.6 virtual bool activemq::commands::BaseDataStructure::equals (const DataStructure ∗value *AMQCPP_UNUSED*) const [inline, virtual]

Referenced by activemq::commands::BooleanExpression::equals(), and activemq::commands::BaseCommand::equals().

6.72.2.7 virtual std::vector<unsigned char> activemq::commands::BaseDataStructure::getMarshaledForm (wireformat::WireFormat ∗wireFormat *AMQCPP_UNUSED*) [inline, virtual]

6.72.2.8 virtual bool activemq::commands::BaseDataStructure::isMarshalAware ( ) const [inline, 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.

Implements **activemq::wireformat::MarshalAware** (p. 1391).

Reimplemented in **activemq::commands::Message** (p. 1421), **activemq::commands::WireFormatInfo** (p. 2244), and **activemq::commands::ActiveMQMapMessage** (p. 216).

6.72.2.9 virtual void activemq::commands::BaseDataStructure::setMarshaledForm (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*, const std::vector< char > &data *AMQCPP_UNUSED*) [inline, virtual]

6.72.2.10 virtual std::string activemq::commands::BaseDataStructure::toString ( ) const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p. 881).

Reimplemented in **activemq::commands::Message** (p. 1424), **activemq::commands::ActiveMQDestination** (p. 198), **activemq::commands::ConsumerInfo** (p. 788), **activemq::commands::MessageId** (p. 1482), **activemq::commands::SessionId** (p. 1845), **activemq::commands::ActiveMQBytesMessage** (p. 138), **activemq::commands::BrokerInfo** (p. 447), **activemq::commands::ConnectionInfo** (p. 755), **activemq::commands::ProducerId** (p. 1694), **activemq::commands::ConnectionId** (p. 747), **activemq::commands::MessageAck** (p. 1451), **activemq::commands::ConsumerId** (p. 779), **activemq::commands::ConsumerControl** (p. 772), **activemq::commands::JournalTopicAck** (p. 1219), **activemq::commands::ProducerInfo** (p. 1700), **activemq::commands::ConnectionControl** (p. 731), **activemq::commands::DestinationInfo** (p. 942), **activemq::commands::MessagePull** (p. 1505), **activemq::commands::SessionInfo** (p. 1851), **activemq::commands::MessageDispatch** (p. 1461), **activemq::commands::MessageDispatchNotification** (p. 1472), **activemq::commands::SubscriptionInfo** (p. 2042), **activemq::commands::XATransactionId** (p. 2281), **activemq::commands::TransactionInfo** (p. 2151), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::JournalQueueAck** (p. 1212), **activemq::commands::JournalTransaction** (p. 1230), **activemq::commands::RemoveSubscriptionInfo** (p. 1766), **activemq::commands::ActiveMQStreamMessage** (p. 286), **activemq::commands::ActiveMQTempDestination** (p. 296), **activemq::commands::LocalTransactionId** (p. 1300), **activemq::commands::NetworkBridgeFilter** (p. 1529), **activemq::commands::ProducerAck** (p. 1685), **activemq::commands::RemoveInfo** (p. 1760), **activemq::commands::ActiveMQMapMessage** (p. 219), **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 865), **activemq::commands::DiscoveryEvent** (p. 951), **activemq::commands::ExceptionResponse** (p. 998), **activemq::commands::PartialCommand** (p. 1610), **activemq::commands::ReplayCommand** (p. 1772), **activemq::commands::BrokerId** (p. 439), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::JournalTrace** (p. 1224), **activemq::commands::Response** (p. 1783), **activemq::commands::FlushCommand** (p. 1070), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::LastPartialCommand** (p. 1247), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::TransactionId** (p. 2145), **activemq::commands::BaseCommand** (p. 386), **activemq::commands::ActiveMQBlobMessage** (p. 125), **activemq::commands::Command** (p. 674), **activemq::commands::WireFormatInfo** (p. 2247), **activemq::commands::ActiveMQQueue** (p. 252), **activemq::commands::ActiveMQTopic** (p. 324), **activemq::commands::ActiveMQTextMessage** (p. 317), **activemq::commands::ActiveMQTempQueue** (p. 303), **activemq::commands::ActiveMQTempTopic** (p. 310), **activemq::commands::ActiveMQObjectMessage** (p. 235), **activemq::commands::ActiveMQMessage** (p. 225), and **activemq::commands::BooleanExpression** (p. 428).

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**

6.73 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 ()

6.73.1 Constructor & Destructor Documentation

6.73.1.1 decaf::net::BindException::BindException ( ) throw () [inline]

Default Constructor.

6.73.1.2 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

6.73.1.3 decaf::net::BindException::BindException (const BindException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.73.1.4 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

6.73.1.5 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.

6.73.1.6 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

6.73.1.7 virtual decaf::net::BindException::~BindException ( ) throw () [inline, virtual]

6.73.2 Member Function Documentation

6.73.2.1 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. 1916).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**BindException.h**

6.74 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, int bufferSize)

  Constructor that initializes the internal buffer.
- virtual ~**BlockingByteArrayInputStream** ()
- virtual void **setByteArray** (**const** unsigned char *buffer, int bufferSize)

- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual void **close** ()

  *Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.*
  The default implementation of this method does nothing.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs while closing the **InputStream** (p. 1134).*

- virtual long long **skip** (long long num)

  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.
  *The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)	*if an I/O error occurs.*
UnsupportedOperation-Exception	*if the concrete stream class does not support skipping bytes.*

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.74.1 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.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ( )

Default Constructor - uses a default internal buffer.

6.74.2.2 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream (const unsigned char * *buffer,* int *bufferSize*)

Constructor that initializes the internal buffer.

See also

setByteArray (p. 416).

6.74.2.3 virtual decaf::io::BlockingByteArrayInputStream::~BlockingByteArrayInputStream ( ) [virtual]

6.74.3 Member Function Documentation

6.74.3.1 virtual int decaf::io::BlockingByteArrayInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.74.3.2 virtual void decaf::io::BlockingByteArrayInputStream::close ( ) [virtual]

Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs while closing the **InputStream** (p. 1134).

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.74.3.3 virtual int decaf::io::BlockingByteArrayInputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.74.3.4 virtual int decaf::io::BlockingByteArrayInputStream::doReadByte ( ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1137).

6.74.3.5 virtual void decaf::io::BlockingByteArrayInputStream::setByteArray (const unsigned char * *buffer,* int *bufferSize*) [virtual]

6.74.3.6 virtual long long decaf::io::BlockingByteArrayInputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1140).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BlockingByteArrayInputStream.h**

6.75 decaf::util::concurrent::BlockingQueue< E > Class Template Reference

A **decaf::util::Queue** (p. 1723) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

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

Inheritance diagram for decaf::util::concurrent::BlockingQueue< E >:

Public Member Functions

- virtual ~**BlockingQueue** ()
- virtual void **put** (**const** E &value)=0

  Inserts the specified element into this queue, waiting if necessary for space to become available.
- virtual bool **offer** (**const** E &e, long long timeout, **const TimeUnit** &unit)=0

  Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
- virtual E **take** ()=0

  Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.
- virtual bool **poll** (E &result, long long timeout, **const TimeUnit** &unit)=0

  Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.
- virtual int **remainingCapacity** () **const** =0

  Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.
- virtual int **drainTo** (**Collection**< E > &c)=0

  Removes all available elements from this queue and adds them to the given collection.
- virtual int **drainTo** (**Collection**< E > &c, int maxElements)=0

  Removes at most the given number of available elements from this queue and adds them to the given collection.

6.75.1 Detailed Description

template<typename E>class decaf::util::concurrent::BlockingQueue< E >

A **decaf::util::Queue** (p. 1723) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

`BlockingQueue` (p. 417) methods come in four forms, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: one throws an exception, the second returns a special value (either `true` or `false`, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:

	Throws exception	*Boolean Flag*	*Blocks*	*Times out*

Insert	**add(e)** (p. 109)	**offer(e)** (p. 420)	**put(e)** (p. 421)	**offer(e, time, unit)** (p. **??**)
Remove	**remove()** (p. 111)	**poll()** (p. 421)	**take()** (p. 422)	**poll(time, unit)** (p. **??**)
Examine	**element()** (p. 111)	**peek()** (p. 1725)	*not applicable*	*not applicable*

A **BlockingQueue** (p. 417) may be capacity bounded. At any given time it may have a `remainingCapacity` beyond which no additional elements can be `put` without blocking. A **BlockingQueue** (p. 417) without any intrinsic capacity constraints always reports a remaining capacity of `Integer::MAX_VALUE`.

BlockingQueue (p. 417) implementations are designed to be used primarily for producer-consumer queues, but additionally support **decaf::util::Collection** (p. 660) interface. So, for example, it is possible to remove an arbitrary element from a queue using `remove(x)`. However, such operations are in general *not* performed very efficiently, and are intended for only occasional use, such as when a queued message is cancelled.

BlockingQueue (p. 417) implementations are thread-safe. All queuing methods achieve their effects atomically using internal locks or other forms of concurrency control. However, the *bulk* **Collection** (p. 660) operations `addAll`, `containsAll`, `retainAll` and `removeAll` are *not* necessarily performed atomically unless specified otherwise in an implementation. So it is possible, for example, for `addAll(c)` to fail (throwing an exception) after adding only some of the elements in `c`.

A **BlockingQueue** (p. 417) does *not* intrinsically support any kind of "close" or "shutdown" operation to indicate that no more items will be added. The needs and usage of such features tend to be implementation-dependent. For example, a common tactic is for producers to insert special *end-of-stream* or *poison* objects, that are interpreted accordingly when taken by consumers.

Usage example, based on a typical producer-consumer scenario. Note that a **BlockingQueue** (p. 417) can safely be used with multiple producers and multiple consumers.

```
 class Producer : public Runnable {
 private:

    BlockingQueue* queue;

public:

    Producer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
         try {
             while( true ) { queue->put( produce() ); }
         } catch( InterruptedException& ex ) { ... handle ...}
     }

    Object produce() { ... }
 }

class Consumer : public Runnable {
 private:

    BlockingQueue* queue;

public:

    Consumer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
         try {
             while( true ) { consume( queue->take() (p. 422) ); }
         } catch( InterruptedException& ex ) { ... handle ...}
     }
```

```
    void consume( Object& x ) { ... }
 }

int main( int argc, char** argv ) {

    BlockingQueue (p. 417) q = new SomeQueueImplementation();
     Producer p( &q );
     Consumer c1( &q );
     Consumer c2( &q );
     Thread t1( &p ).start();
     Thread t2( &c1 ).start();
     Thread t3( &c2 ).start();
 }
```

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a **BlockingQueue** (p. 417) *happen-before* actions subsequent to the access or removal of that element from the **BlockingQueue** (p. 417) in another thread.

Since

1.0

6.75.2 Constructor & Destructor Documentation

6.75.2.1 template<typename E> virtual decaf::util::concurrent::BlockingQueue< E >::~BlockingQueue ( ) [inline, virtual]

6.75.3 Member Function Documentation

6.75.3.1 template<typename E> virtual int decaf::util::concurrent::BlockingQueue< E >::drainTo (Collection< E > & *c*) [pure virtual]

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c	the collection to transfer elements into

Returns

the number of elements transferred

Exceptions

UnsupportedOperation-Exception	if addition of elements is not supported by the specified collection
IllegalArgumentException	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1260), and **decaf::util::concurrent::-SynchronousQueue< E >** (p. 2060).

6.75.3.2 template<typename E> virtual int decaf::util::concurrent::BlockingQueue< E >::drainTo (Collection< E > & *c*, int *maxElements*) [pure virtual]

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c	the collection to transfer elements into
maxElements	the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

UnsupportedOperationException	if addition of elements is not supported by the specified collection
IllegalArgumentException	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1261), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2060).

6.75.3.3 template<typename E> virtual bool decaf::util::concurrent::BlockingQueue< E >::offer (const E & *e*, long long *timeout*, const TimeUnit & *unit*) [pure virtual]

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Parameters

e	the element to add
timeout	how long to wait before giving up, in units of `unit`
unit	a **`TimeUnit`** (p. 2134) determining how to interpret the `timeout` parameter

Returns

`true` if successful, or `false` if the specified waiting time elapses before space is available

Exceptions

InterruptedException	if interrupted while waiting
NullPointerException	if the specified element is null
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1262), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2062).

6.75.3.4 template<typename E> virtual bool decaf::util::concurrent::BlockingQueue< E >::poll (E & *result,* long long *timeout,* const TimeUnit & *unit*) [pure virtual]

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

Parameters

result	the referenced value that will be assigned the value retrieved from the **Queue** (p. 1723). Undefined if this methods returned false.
timeout	how long to wait before giving up, in units of `unit`
unit	a **`TimeUnit`** (p. 2134) determining how to interpret the `timeout` parameter.

Returns

`true` if successful or `false` if the specified waiting time elapses before an element is available.

Exceptions

InterruptedException	if interrupted while waiting

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1263), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2062).

6.75.3.5 template<typename E> virtual void decaf::util::concurrent::BlockingQueue< E >::put (const E & *value*) [pure virtual]

Inserts the specified element into this queue, waiting if necessary for space to become available.

Parameters

value	the element to add

Exceptions

InterruptedException	if interrupted while waiting
NullPointerException	if the specified element is null
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1264), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2063).

6.75.3.6 template<typename E> virtual int decaf::util::concurrent::BlockingQueue< E >::remainingCapacity ( ) const [pure virtual]

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

Returns

the remaining capacity

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1265), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2063).

6.75.3.7 template<typename E> virtual E decaf::util::concurrent::BlockingQueue< E >::take ( ) [pure virtual]

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Returns

the head of this queue

Exceptions

InterruptedException	if interrupted while waiting

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1266), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2064).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**BlockingQueue.h**

6.76 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. 422) 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. 422) 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** (p. 2031) passed and extracts an bool.*
- static std::string **toString** (bool value)

  *Converts the bool to a **String** (p. 2031) 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.

6.76.1 Constructor & Destructor Documentation

6.76.1.1 decaf::lang::Boolean::Boolean (bool *value*)

Parameters

value	- primitive boolean to wrap.

6.76.1.2 decaf::lang::Boolean::Boolean (const std::string & *value*)

Parameters

value	- **String** (p. 2031) value to convert to a boolean.

6.76.1.3 virtual decaf::lang::Boolean::~Boolean ( ) [inline, virtual]

6.76.2 Member Function Documentation

6.76.2.1 bool decaf::lang::Boolean::booleanValue ( ) const [inline]

Returns

the primitive boolean value of this object

6.76.2.2 virtual int decaf::lang::Boolean::compareTo (const Boolean & *b*) const [virtual]

Compares this **Boolean** (p. 422) instance with another.

Parameters

b	- the **Boolean** (p. 422) 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**< **Boolean** > (p. 687).

6.76.2.3 virtual int decaf::lang::Boolean::compareTo (const bool & *b*) const [virtual]

Compares this **Boolean** (p. 422) instance with another.

Parameters

b	- the **Boolean** (p. 422) 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. 687).

6.76.2.4 bool decaf::lang::Boolean::equals (const Boolean & *b*) const [inline, virtual]

Returns

true if the two **Boolean** (p. 422) Objects have the same value.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 688).

6.76.2.5 bool decaf::lang::Boolean::equals (const bool & *b*) const [inline, virtual]

Returns

true if the two **Boolean** (p. 422) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 688).

6.76.2.6 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.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 688).

6.76.2.7 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. 688).

6.76.2.8 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.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 688).

6.76.2.9 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. 688).

6.76.2.10 static bool decaf::lang::Boolean::parseBoolean (const std::string & *value*) [static]

Parses the **String** (p. 2031) passed and extracts an bool.

Parameters

value	The std::string value to parse

Returns

bool value

6.76.2.11 std::string decaf::lang::Boolean::toString ( ) const

Returns

the string representation of this Booleans value.

6.76.2.12 static std::string decaf::lang::Boolean::toString (bool *value*) [static]

Converts the bool to a **String** (p. 2031) representation.

Parameters

value	The bool value to convert.

Returns

std::string representation of the bool value passed.

6.76.2.13 static Boolean decaf::lang::Boolean::valueOf (bool *value*) [static]

Parameters

value	The bool value to convert to a **Boolean** (p. 422) instance.

Returns

a **Boolean** (p. 422) instance of the primitive boolean value

6.76.2.14 static Boolean decaf::lang::Boolean::valueOf (const std::string & *value*) [static]

Parameters

value	The std::string value to convert to a **Boolean** (p. 422) instance.

Returns

a **Boolean** (p. 422) instance of the string value

6.76.3 Field Documentation

6.76.3.1 const Boolean decaf::lang::Boolean::_FALSE [static]

The Class object representing the primitive false boolean.

6.76.3.2 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**

6.77 activemq::commands::BooleanExpression Class Reference

`#include <src/main/activemq/commands/BooleanExpression.h>`

Inheritance diagram for activemq::commands::BooleanExpression:

Public Member Functions

- **BooleanExpression** ()
- virtual ∼**BooleanExpression** () throw ()
- virtual **DataStructure** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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)
- virtual std::string **toString** () **const**

  *Returns a string containing the information for this **DataStructure*** (p. *877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure*** (p. *877) passed in to this one, and returns if they are equivalent.*

6.77.1 Constructor & Destructor Documentation

6.77.1.1 activemq::commands::BooleanExpression::BooleanExpression ( ) [inline]

6.77.1.2 virtual activemq::commands::BooleanExpression::∼BooleanExpression ( ) throw () [inline, virtual]

6.77.2 Member Function Documentation

6.77.2.1 virtual DataStructure∗ activemq::commands::BooleanExpression::cloneDataStructure ( ) const [inline, virtual]

Clone this obbject 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. 878).

6.77.2.2 virtual void activemq::commands::BooleanExpression::copyDataStructure (const DataStructure ∗src *AMQCPP_UNUSED*) [inline, virtual]

Reimplemented from **activemq::commands::BaseDataStructure** (p. 410).

6.77.2.3 virtual bool activemq::commands::BooleanExpression::equals (const DataStructure ∗ *value*) const [inline, virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

References activemq::commands::BaseDataStructure::equals().

6.77.2.4 virtual std::string activemq::commands::BooleanExpression::toString ( ) const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

References activemq::commands::BaseDataStructure::toString().

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BooleanExpression.h**

6.78 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** () throw ()
- bool **readBoolean** ()

  Read a boolean data element from the internal data buffer.
- void **writeBoolean** (bool value)

  Writes a Boolean value to the internal data buffer.
- void **marshal** (**decaf::io::DataOutputStream** *dataOut)

  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)

  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.

6.78.1 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.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ( )

6.78.2.2 virtual activemq::wireformat::openwire::utils::BooleanStream::~BooleanStream ( ) throw () [virtual]

6.78.3 Member Function Documentation

6.78.3.1 void activemq::wireformat::openwire::utils::BooleanStream::clear ( )

Clears to old position markers, data starts at the beginning.

6.78.3.2 void activemq::wireformat::openwire::utils::BooleanStream::marshal (decaf::io::DataOutputStream * *dataOut*)

Marshal the data to a DataOutputStream.

Parameters

dataOut	- Stream to write the data to.

Exceptions

IOException	if an I/O error occurs during this operation.

6.78.3.3 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.

Exceptions

IOException	if an I/O error occurs during this operation.

6.78.3.4 int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize ( )

Calc the size that data is marshalled to.

Returns

int size of marshalled data.

6.78.3.5 bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean ( )

Read a boolean data element from the internal data buffer.

Returns

boolean from the stream

Exceptions

IOException	if an I/O error occurs during this operation.

Referenced by activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2().

6.78.3.6 void activemq::wireformat::openwire::utils::BooleanStream::unmarshal (decaf::io::DataInputStream * *dataIn*)

Unmarshal a Boolean data stream from the Input Stream.

Parameters

dataIn	- Input Stream to read data from.

Exceptions

IOException	if an I/O error occurs during this operation.

6.78.3.7 void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean (bool *value*)

Writes a Boolean value to the internal data buffer.

Parameters

value	- boolean data to write.

Exceptions

IOException	if an I/O error occurs during this operation.

Referenced by activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1().

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**BooleanStream.h**

6.79 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 ()

6.79.1 Constructor & Destructor Documentation

6.79.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( ) throw () [inline]

Default Constructor.

6.79.1.2 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

6.79.1.3 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const BrokenBarrierException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The Exception to copy in this new instance.

6.79.1.4 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.

6.79.1.5 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

6.79.1.6 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

6.79.1.7 virtual decaf::util::concurrent::BrokenBarrierException::~BrokenBarrierException ( ) throw () [virtual]

6.79.2 Member Function Documentation

6.79.2.1 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. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**BrokenBarrierException.h**

6.80 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. 877) 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)

  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. 1412).*
- 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.

6.80.1 Detailed Description

This class represents an Exception sent from the Broker.

The Broker sends java Throwables, so we must mimic its structure here.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 activemq::commands::BrokerError::BrokerError ( )

6.80.2.2 virtual activemq::commands::BrokerError::~BrokerError ( ) [virtual]

6.80.3 Member Function Documentation

6.80.3.1 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. 878).

References copyDataStructure().

6.80.3.2 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. 382).

Referenced by cloneDataStructure().

6.80.3.3 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

6.80.3.4 virtual unsigned char activemq::commands::BrokerError::getDataStructureType ( ) const [inline, virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.80.3.5 virtual const std::string& activemq::commands::BrokerError::getExceptionClass ( ) const [inline, virtual]

Gets the string holding the Exception Class name.

Returns

Exception Class name

6.80.3.6 virtual const std::string& activemq::commands::BrokerError::getMessage ( ) const [inline, virtual]

Gets the string holding the error message.

Returns

String **Message** (p. 1412)

6.80.3.7 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

6.80.3.8 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

6.80.3.9 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

6.80.3.10 virtual void activemq::commands::BrokerError::setMessage (const std::string & *message*) [inline, virtual]

Sets the string that contains the error **Message** (p. 1412).

Parameters

message	- String Error **Message** (p. 1412)

6.80.3.11 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

stackTrace-Elements	- Stack Trace Elements

6.80.3.12 virtual decaf::lang::Pointer<commands::Command> activemq::commands::BrokerError::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

6.81 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 ()

6.81.1 Constructor & Destructor Documentation

6.81.1.1 activemq::exceptions::BrokerException::BrokerException ( ) throw () [inline]

6.81.1.2 activemq::exceptions::BrokerException::BrokerException (const exceptions::ActiveMQException & *ex*) throw () [inline]

6.81.1.3 activemq::exceptions::BrokerException::BrokerException (const BrokerException & *ex*) throw () [inline]

6.81.1.4 activemq::exceptions::BrokerException::BrokerException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

6.81.1.5 activemq::exceptions::BrokerException::BrokerException (const char * *file*, const int *lineNumber*, const commands::BrokerError * *error*) throw () [inline]

References decaf::lang::Exception::getMessage().

6.81.1.6 virtual activemq::exceptions::BrokerException::~BrokerException ( ) throw () [inline, virtual]

6.81.2 Member Function Documentation

6.81.2.1 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. 204).

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**BrokerException.h**

6.82 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **BrokerId** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *__value__) **const**

  *Compares the **DataStructure** (p. 877) 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**

6.82.1 Member Typedef Documentation

6.82.1.1 typedef decaf::lang::PointerComparator<BrokerId> activemq::commands::BrokerId::COMPARATOR

6.82.2 Constructor & Destructor Documentation

6.82.2.1 activemq::commands::BrokerId::BrokerId ( )

6.82.2.2 activemq::commands::BrokerId::BrokerId (const BrokerId & *other*)

6.82.2.3 virtual activemq::commands::BrokerId::~BrokerId ( ) [virtual]

6.82.3 Member Function Documentation

6.82.3.1 virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.82.3.2 virtual int activemq::commands::BrokerId::compareTo (const BrokerId & *value*) const [virtual]

6.82.3.3 virtual void activemq::commands::BrokerId::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. 879).

6.82.3.4 virtual bool activemq::commands::BrokerId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.82.3.5 virtual bool activemq::commands::BrokerId::equals (const BrokerId & *value*) const [virtual]

6.82.3.6 virtual unsigned char activemq::commands::BrokerId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.82.3.7 virtual const std::string& activemq::commands::BrokerId::getValue ( ) const [virtual]

6.82.3.8 virtual std::string& activemq::commands::BrokerId::getValue ( ) [virtual]

6.82.3.9 virtual bool activemq::commands::BrokerId::operator< (const BrokerId & *value*) const [virtual]

6.82.3.10 BrokerId& activemq::commands::BrokerId::operator= (const BrokerId & *other*)

6.82.3.11 virtual bool activemq::commands::BrokerId::operator== (const BrokerId & *value*) const [virtual]

6.82.3.12 virtual void activemq::commands::BrokerId::setValue (const std::string & *value*) [virtual]

6.82.3.13 virtual std::string activemq::commands::BrokerId::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.82.4 Field Documentation

6.82.4.1 const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124 [static]

6.82.4.2 std::string activemq::commands::BrokerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerId.h**

6.83 activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 440).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/BrokerIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual ~**BrokerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.83.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 440).

NOTE!: This file is 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::generated::BrokerIdMarshaller::BrokerIdMarshaller ( ) [inline]

6.83.2.2 virtual activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::~BrokerIdMarshaller ( ) [inline, virtual]

6.83.3 Member Function Documentation

6.83.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.83.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.83.3.3 virtual void activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.83.3.4 virtual void activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.83.3.5 virtual int activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.83.3.6 virtual void activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.83.3.7 virtual void activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**BrokerIdMarshaller.h**

6.84 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **BrokerInfo** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 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**

6.84.1 Constructor & Destructor Documentation

6.84.1.1 activemq::commands::BrokerInfo::BrokerInfo ( )

6.84.1.2 virtual activemq::commands::BrokerInfo::~BrokerInfo ( ) [virtual]

6.84.2 Member Function Documentation

6.84.2.1 virtual BrokerInfo* activemq::commands::BrokerInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.84.2.2 virtual void activemq::commands::BrokerInfo::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. 382).

6.84.2.3 virtual bool activemq::commands::BrokerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.84.2.4 virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ( ) const [virtual]

6.84.2.5 virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ( ) [virtual]

6.84.2.6 virtual const std::string& activemq::commands::BrokerInfo::getBrokerName ( ) const [virtual]

6.84.2.7 virtual std::string& activemq::commands::BrokerInfo::getBrokerName ( ) [virtual]

6.84.2.8 virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ( ) const [virtual]

6.84.2.9 virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ( ) [virtual]

6.84.2.10 virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL ( ) const [virtual]

6.84.2.11 virtual std::string& activemq::commands::BrokerInfo::getBrokerURL ( ) [virtual]

6.84.2.12 virtual long long activemq::commands::BrokerInfo::getConnectionId ( ) const [virtual]

6.84.2.13 virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.84.2.14 virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties ( ) const [virtual]

6.84.2.15 virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties ( ) [virtual]

6.84.2.16 virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ( ) const [virtual]

6.84.2.17 virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ( ) [virtual]

6.84.2.18 virtual bool activemq::commands::BrokerInfo::isBrokerInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isBrokerInfo()** (p. 446) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 383).

6.84.2.19 virtual bool activemq::commands::BrokerInfo::isDuplexConnection ( ) const [virtual]

6.84.2.20 virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration ( ) const [virtual]

6.84.2.21 virtual bool activemq::commands::BrokerInfo::isMasterBroker ( ) const [virtual]

6.84.2.22 virtual bool activemq::commands::BrokerInfo::isNetworkConnection ( ) const [virtual]

6.84.2.23 virtual bool activemq::commands::BrokerInfo::isSlaveBroker ( ) const [virtual]

6.84.2.24 virtual void activemq::commands::BrokerInfo::setBrokerId (const Pointer< BrokerId > & *brokerId*) [virtual]

6.84.2.25 virtual void activemq::commands::BrokerInfo::setBrokerName (const std::string & *brokerName*) [virtual]

6.84.2.26 virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl (const std::string & *brokerUploadUrl*) [virtual]

6.84.2.27 virtual void **activemq::commands::BrokerInfo::setBrokerURL** (const std::string & *brokerURL*) [virtual]

6.84.2.28 virtual void **activemq::commands::BrokerInfo::setConnectionId** (long long *connectionId*) [virtual]

6.84.2.29 virtual void **activemq::commands::BrokerInfo::setDuplexConnection** (bool *duplexConnection*) [virtual]

6.84.2.30 virtual void **activemq::commands::BrokerInfo::setFaultTolerantConfiguration** (bool *faultTolerantConfiguration*) [virtual]

6.84.2.31 virtual void **activemq::commands::BrokerInfo::setMasterBroker** (bool *masterBroker*) [virtual]

6.84.2.32 virtual void **activemq::commands::BrokerInfo::setNetworkConnection** (bool *networkConnection*) [virtual]

6.84.2.33 virtual void **activemq::commands::BrokerInfo::setNetworkProperties** (const std::string & *networkProperties*) [virtual]

6.84.2.34 virtual void **activemq::commands::BrokerInfo::setPeerBrokerInfos** (const std::vector< decaf::lang::Pointer< BrokerInfo > > & *peerBrokerInfos*) [virtual]

6.84.2.35 virtual void **activemq::commands::BrokerInfo::setSlaveBroker** (bool *slaveBroker*) [virtual]

6.84.2.36 virtual std::string **activemq::commands::BrokerInfo::toString** ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.84.2.37 virtual Pointer<Command> **activemq::commands::BrokerInfo::visit** (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.84.3 Field Documentation

6.84.3.1 Pointer<BrokerId> **activemq::commands::BrokerInfo::brokerId** [protected]

6.84.3.2 std::string **activemq::commands::BrokerInfo::brokerName** [protected]

6.84.3.3 std::string **activemq::commands::BrokerInfo::brokerUploadUrl** [protected]

6.84.3.4 std::string activemq::commands::BrokerInfo::brokerURL [protected]

6.84.3.5 long long activemq::commands::BrokerInfo::connectionId [protected]

6.84.3.6 bool activemq::commands::BrokerInfo::duplexConnection [protected]

6.84.3.7 bool activemq::commands::BrokerInfo::faultTolerantConfiguration [protected]

6.84.3.8 const unsigned char activemq::commands::BrokerInfo::ID_BROKERINFO = 2 [static]

6.84.3.9 bool activemq::commands::BrokerInfo::masterBroker [protected]

6.84.3.10 bool activemq::commands::BrokerInfo::networkConnection [protected]

6.84.3.11 std::string activemq::commands::BrokerInfo::networkProperties [protected]

6.84.3.12 std::vector< decaf::lang::Pointer<BrokerInfo> > activemq::commands::BrokerInfo::peerBrokerInfos [protected]

6.84.3.13 bool activemq::commands::BrokerInfo::slaveBroker [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerInfo.h**

6.85 activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 448).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/BrokerInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual ~**BrokerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.85.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 448).

NOTE!: This file is 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::generated::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]

6.85.2.2 virtual activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::∼BrokerInfoMarshaller ( ) [inline, virtual]

6.85.3 Member Function Documentation

6.85.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.85.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.85.3.3 virtual void activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::looseMarshal (OpenWireFormat ∗ *format*, commands::DataStructure ∗ *command*, decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.85.3.4 virtual void activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.85.3.5 virtual int activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.85.3.6 virtual void activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.85.3.7 virtual void activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**BrokerInfoMarshaller.h**

6.86 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** (int capactiy)
- **Buffer** (**const Buffer** &other)
- virtual ~**Buffer** ()
- virtual int **capacity** () **const**
- virtual int **position** () **const**
- virtual **Buffer** & **position** (int newPosition)

  Sets this buffer's position.
- virtual int **limit** () **const**
- virtual **Buffer** & **limit** (int newLimit)

  Sets this buffer's limit.
- virtual **Buffer** & **mark** ()

  Sets this buffer's mark at its position.

- virtual **Buffer** & **reset** ()

  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 int **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

- int **_position**
- int **_capacity**
- int **_limit**
- int **_mark**
- bool **_markSet**

6.86.1 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. 474) and a relative put operation throws a **BufferOverflowException** (p. 472); 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. 1193) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values: 0 <= mark <= position <= limit <= 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. 454) 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. 454) 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. 456) 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. 1739) 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);

6.86.2 Constructor & Destructor Documentation

6.86.2.1 decaf::nio::Buffer::Buffer (int *capactiy*)

6.86.2.2 decaf::nio::Buffer::Buffer (const Buffer & *other*)

6.86.2.3 virtual decaf::nio::Buffer::~Buffer ( ) [inline, virtual]

6.86.3 Member Function Documentation

6.86.3.1 virtual int decaf::nio::Buffer::capacity ( ) const [inline, virtual]

Returns

this buffer's capacity.

6.86.3.2 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.

6.86.3.3 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.

6.86.3.4 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.

6.86.3.5 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.

Implemented in **decaf::nio::ByteBuffer** (p. 548), **decaf::internal::nio::ByteArrayBuffer** (p. 521), **decaf::internal::nio::CharArrayBuffer** (p. 607), **decaf::internal::nio::DoubleArrayBuffer** (p. 973), **decaf::internal::nio::FloatArrayBuffer** (p. 1057), **decaf::internal::nio::IntArrayBuffer** (p. 1151), **decaf::internal::nio::LongArrayBuffer** (p. 1358), and **decaf::internal::nio::ShortArrayBuffer** (p. 1873).

6.86.3.6 virtual int decaf::nio::Buffer::limit ( ) const [inline, virtual]

Returns

this buffers Limit

6.86.3.7 virtual Buffer& decaf::nio::Buffer::limit (int *newLimit*) [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.

6.86.3.8 virtual Buffer& decaf::nio::Buffer::mark ( ) [virtual]

Sets this buffer's mark at its position.

Returns

a reference to this buffer.

6.86.3.9 virtual int decaf::nio::Buffer::position ( ) const [inline, virtual]

Returns

the current position in the buffer

6.86.3.10 virtual Buffer& decaf::nio::Buffer::position (int *newPosition*) [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 postion in the buffer to set.

Returns

a reference to This buffer.

Exceptions

IllegalArgumentException	if preconditions on the new pos don't hold.

6.86.3.11 virtual int 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

6.86.3.12 virtual Buffer& decaf::nio::Buffer::reset ( ) [virtual]

Resets this buffer's position to the previously-marked position.

Returns

a reference to this buffer.

Exceptions

InvalidMarkException *(p. 1193)*	- If the mark has not been set

6.86.3.13 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.

6.86.4 Field Documentation

6.86.4.1 int decaf::nio::Buffer::_capacity [protected]

6.86.4.2 int decaf::nio::Buffer::_limit [protected]

6.86.4.3 int decaf::nio::Buffer::_mark [protected]

6.86.4.4 bool decaf::nio::Buffer::_markSet [protected]

6.86.4.5 int decaf::nio::Buffer::_position [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**Buffer.h**

6.87 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, int bufferSize, bool **own**=false)

  Constructor.
- virtual ∼**BufferedInputStream** ()
- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual void **close** ()

  *Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.*
  The default implementation of this method does nothing.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs while closing the **InputStream** (p. 1134).*

- virtual long long **skip** (long long num)

  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.
  *The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)	*if an I/O error occurs.*
UnsupportedOperation-Exception	*if the concrete stream class does not support skipping bytes.*

- 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.
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.
Calling mark on a closed stream instance should have no effect.
The default implementation of this method does nothing.

Parameters

readLimit	*The max bytes read before marked position is invalid.*

- virtual void **reset** ()

  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. 1198) might be thrown.*
  - *If such an **IOException** (p. 1198) 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. 1198).*
  - *If an **IOException** (p. 1198) 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.*

  *The default implementation of this method throws an **IOException** (p. 1198).*

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual bool **markSupported** () **const**

  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.
  The default implementation of this method returns false.

  Returns

  true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.87.1 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.

6.87.2 Constructor & Destructor Documentation

6.87.2.1 decaf::io::BufferedInputStream::BufferedInputStream (InputStream * *stream,* bool *own* = `false`)

Constructor.

Parameters

stream	The target input stream to buffer.
own	Indicates if we own the stream object, defaults to false.

6.87.2.2 decaf::io::BufferedInputStream::BufferedInputStream (InputStream * *stream,* int *bufferSize,* bool *own* = false)

Constructor.

Parameters

stream	The target input stream to buffer.
bufferSize	The size in bytes to allocate for the internal buffer.
own	Indicates if we own the stream object, defaults to false.

Exceptions

IllegalArgumentException	is the size is zero or negative.

6.87.2.3 virtual decaf::io::BufferedInputStream::~BufferedInputStream ( ) [virtual]

6.87.3 Member Function Documentation

6.87.3.1 virtual int decaf::io::BufferedInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1034).

6.87.3.2 virtual void decaf::io::BufferedInputStream::close ( ) [virtual]

Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs while closing the **InputStream** (p. 1134).

Reimplemented from **decaf::io::FilterInputStream** (p. 1034).

6.87.3.3 virtual int decaf::io::BufferedInputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.87.3.4 virtual int decaf::io::BufferedInputStream::doReadByte ( ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.87.3.5 virtual void decaf::io::BufferedInputStream::mark (int *readLimit*) [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.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.87.3.6 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.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.87.3.7 virtual void decaf::io::BufferedInputStream::reset ( ) [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. 1198) might be thrown.
- If such an **IOException** (p. 1198) 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. 1198).
- If an **IOException** (p. 1198) 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.

The default implementation of this method throws an **IOException** (p. 1198).

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1036).

6.87.3.8 virtual long long decaf::io::BufferedInputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::FilterInputStream** (p. 1036).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedInputStream.h**

6.88 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, int bufferSize, bool **own**=false)
  Constructor.
- virtual ∼**BufferedOutputStream** ()
- virtual void **flush** ()

inheritDoc}

- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArray** (**const** unsigned char *buffer, int size)
- virtual void **doWriteArrayBounded** (**const** unsigned char *buffer, int size, int offset, int length)

6.88.1 Detailed Description

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

6.88.2 Constructor & Destructor Documentation

6.88.2.1 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.

6.88.2.2 decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * *stream,* int *bufferSize,* bool *own* = `false`)

Constructor.

Parameters

stream	The target output stream.
bufferSize	The size for the internal buffer.
own	Indicates if this class owns the stream pointer.

Exceptions

IllegalArgumentException	if the bufferSize given is negative.

6.88.2.3 virtual decaf::io::BufferedOutputStream::~BufferedOutputStream ( ) `[virtual]`

6.88.3 Member Function Documentation

6.88.3.1 virtual void decaf::io::BufferedOutputStream::doWriteArray (const unsigned char * *buffer,* int *size*) `[protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.88.3.2 virtual void decaf::io::BufferedOutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) `[protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.88.3.3 virtual void decaf::io::BufferedOutputStream::doWriteByte (unsigned char *c*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.88.3.4 virtual void decaf::io::BufferedOutputStream::flush ( ) [virtual]

inheritDoc}

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedOutputStream.h**

6.89 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p. 74) 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** (int 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, int size, int offset, int length)

  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** (int 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, int size, int offset, int length)

  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** (int 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, int size, int offset, int length)

  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** (int 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, int size, int offset, int length)

  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** (int 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, int size, int offset, int length)

  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** (int 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, int size, int offset, int length)

  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** (int 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, int size, int offset, int length)

  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.

6.89.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p. 74) package to create the various default version of the NIO interfaces.

Since

1.0

6.89.2 Constructor & Destructor Documentation

6.89.2.1 virtual decaf::internal::nio::BufferFactory::~BufferFactory ( ) [inline, virtual]

6.89.3 Member Function Documentation

6.89.3.1 static decaf::nio::ByteBuffer∗ decaf::internal::nio::BufferFactory::createByteBuffer (int *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.

Exceptions

IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.2 static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [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.
size	The size of the specified 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 the returned pointer.

Exceptions

NullPointerException	if the buffer given in Null.
IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.3 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.

6.89.3.4 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (int *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.

Exceptions

IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.5 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (char * *buffer,* int *size,* int *offset,* int *length*) [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.
size	The size of the specified 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 the returned pointer.

Exceptions

NullPointerException	if the buffer given in Null.
IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.6 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.

6.89.3.7 static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (int *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.

Exceptions

IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.8 static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (double * *buffer,* int *size,* int *offset,* int *length*) [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.
size	The size of the specified 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 the returned pointer.

Exceptions

NullPointerException	if the buffer given in Null.
IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.9 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.

6.89.3.10 static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (int *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 FloatBuffer which the caller owns.

Exceptions

IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.11 static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (float * *buffer,* int *size,* int *offset,* int *length*) [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.
size	The size of the specified buffer.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new FloatBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException	if the buffer given in Null.
IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.12 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 FloatBuffer that is backed by buffer, caller owns.

6.89.3.13 static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (int *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 IntBuffer which the caller owns.

Exceptions

IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.14 static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (int * *buffer,* int *size,* int *offset,* int *length*) [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.
size	The size of the specified buffer.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new IntBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException	if the buffer given in Null.
IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.15 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 IntBuffer that is backed by buffer, caller owns.

6.89.3.16 static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (int *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.

Exceptions

IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.17 static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (long long * *buffer,* int *size,* int *offset,* int *length*) [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.
size	The size of the specified buffer.
offset	The offset of the subarray to be used.

length	The length of the subarray to be used.

Returns

a new LongBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException	if the buffer given in Null.
IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.18 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 LongBuffer that is backed by buffer, caller owns.

6.89.3.19 static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (int *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 ShortBuffer which the caller owns.

Exceptions

IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.20 static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (short * *buffer*, int *size*, int *offset*, int *length*) [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.
size	The size of the specified buffer.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new ShortBuffer that is backed by buffer, caller owns the returned pointer.

Exceptions

NullPointerException	if the buffer given in Null.
IndexOutOfBounds-Exception	if the capacity specified is negative.

6.89.3.21 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**

6.90 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 ()

6.90.1 Constructor & Destructor Documentation

6.90.1.1 decaf::nio::BufferOverflowException::BufferOverflowException ( ) throw () [inline]

Default Constructor.

6.90.1.2 decaf::nio::BufferOverflowException::BufferOverflowException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.90.1.3 decaf::nio::BufferOverflowException::BufferOverflowException (const BufferOverflowException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.90.1.4 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

6.90.1.5 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.

6.90.1.6 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

6.90.1.7 virtual decaf::nio::BufferOverflowException::~BufferOverflowException ( ) throw () [inline, virtual]

6.90.2 Member Function Documentation

6.90.2.1 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. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferOverflowException.h**

6.91 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 ()

6.91.1 Constructor & Destructor Documentation

6.91.1.1 decaf::nio::BufferUnderflowException::BufferUnderflowException ( ) throw () [inline]

Default Constructor.

6.91.1.2 decaf::nio::BufferUnderflowException::BufferUnderflowException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.91.1.3 decaf::nio::BufferUnderflowException::BufferUnderflowException (const BufferUnderflowException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.91.1.4 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

6.91.1.5 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.

6.91.1.6 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

6.91.1.7 virtual decaf::nio::BufferUnderflowException::~BufferUnderflowException ( ) throw () [inline, virtual]

6.91.2 Member Function Documentation

6.91.2.1 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. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferUnderflowException.h**

6.92 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)

  *Creates a new **Byte** (p. 476) instance from the given string.*
- virtual ~**Byte** ()

- virtual int **compareTo** (**const Byte** &c) **const**

  *Compares this **Byte** (p. 476) 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. 476) 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)

  *Decodes a **String** (p. 2031) into a **Byte** (p. 476).*
- static unsigned char **parseByte** (**const** std::string &s, int radix)

  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)

  Parses the string argument as a signed decimal unsigned char.
- static **Byte valueOf** (unsigned char value)

  *Returns a **Character** (p. 593) instance representing the specified char value.*
- static **Byte valueOf** (**const** std::string &value)

  *Returns a **Byte** (p. 476) object holding the value given by the specified std::string.*
- static **Byte valueOf** (**const** std::string &value, int radix)

  *Returns a **Byte** (p. 476) 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.

6.92.1 Constructor & Destructor Documentation

6.92.1.1 decaf::lang::Byte::Byte (unsigned char *value*)

Parameters

value	- the primitive value to wrap

6.92.1.2 decaf::lang::Byte::Byte (const std::string & *value*)

Creates a new **Byte** (p. 476) instance from the given string.

Parameters

value	The string to convert to an unsigned char

Exceptions

NumberFormatException	if the string is not a valid byte.

6.92.1.3 virtual decaf::lang::Byte::~Byte ( ) [inline, virtual]

6.92.2 Member Function Documentation

6.92.2.1 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.

Reimplemented from **decaf::lang::Number** (p. 1544).

6.92.2.2 virtual int decaf::lang::Byte::compareTo (const Byte & *c*) const [inline, virtual]

Compares this **Byte** (p. 476) instance with another.

Parameters

c	- the **Byte** (p. 476) 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 repesents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Byte** > (p. 687).

6.92.2.3 virtual int decaf::lang::Byte::compareTo (const unsigned char & *c*) const [inline, virtual]

Compares this **Byte** (p. 476) 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 repesents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **unsigned char** > (p. 687).

6.92.2.4 static Byte decaf::lang::Byte::decode (const std::string & *value*) [static]

Decodes a **String** (p. 2031) into a **Byte** (p. 476).

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. 481) 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** (p. 2031) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value	- The string to decode

Returns

a **Byte** (p. 476) object containing the decoded value

Exceptions

NumberFomatException	if the string is not formatted correctly.

6.92.2.5 virtual double decaf::lang::Byte::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.92.2.6 bool decaf::lang::Byte::equals (const Byte & *c*) const [inline, virtual]

Returns

true if the two **Byte** (p. 476) Objects have the same value.

Implements **decaf::lang::Comparable**< **Byte** > (p. 688).

6.92.2.7 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. 688).

6.92.2.8 virtual float decaf::lang::Byte::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.92.2.9 virtual int decaf::lang::Byte::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.92.2.10 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.

Implements **decaf::lang::Number** (p. 1545).

6.92.2.11 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.

Implements **decaf::lang::Comparable< Byte >** (p. 688).

6.92.2.12 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. 688).

6.92.2.13 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.

Implements **decaf::lang::Comparable< Byte >** (p. 688).

6.92.2.14 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. 688).

6.92.2.15 static unsigned char decaf::lang::Byte::parseByte (const std::string & *s,* int *radix*) [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. 595) 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. 599) or larger than **Character::MAX_RADIX** (p. 599).
- 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** (p. 2031) 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** (p. 2031) does not contain a parsable unsigned char.

6.92.2.16 static unsigned char decaf::lang::Byte::parseByte (const std::string & *s*) [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** (p. 2031) to convert to a unsigned char

Returns

the converted unsigned char value

Exceptions

NumberFormatException	if the string is not a unsigned char.

6.92.2.17 virtual short decaf::lang::Byte::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1545).

6.92.2.18 std::string decaf::lang::Byte::toString ( ) const

Returns

this **Byte** (p. 476) Object as a **String** (p. 2031) Representation

6.92.2.19 static std::string decaf::lang::Byte::toString (unsigned char *value*) [static]

Returns

a string representing the primitive value as Base 10

6.92.2.20 static Byte decaf::lang::Byte::valueOf (unsigned char *value*) [inline, static]

Returns a **Character** (p. 593) instance representing the specified char value.

Parameters

value	- the primitive char to wrap.

Returns

a new **Character** (p. 593) instance that wraps this value.

6.92.2.21 static Byte decaf::lang::Byte::valueOf (const std::string & *value*) [static]

Returns a **Byte** (p. 476) 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. 476) object that represents the unsigned char value specified by the string.

Parameters

value	- std::string to parse as base 10

Returns

new **Byte** (p. 476) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a decimal unsigned char.

6.92.2.22 static Byte decaf::lang::Byte::valueOf (const std::string & *value,* int *radix*) [static]

Returns a **Byte** (p. 476) 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. 476) 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. 476) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a valid unsigned char.

6.92.3 Field Documentation

6.92.3.1 const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80 [static]

The maximum value that a unsigned char can take on.

6.92.3.2 const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F [static]

The minimum value that a unsigned char can take on.

6.92.3.3 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**

6.93 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>`

Data Structures

- union **Array**

Public Member Functions

- **ByteArrayAdapter** (int size)

  Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.
- **ByteArrayAdapter** (unsigned char ∗array, int size, bool own=false)

  Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (char ∗array, int size, bool own=false)

  Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (double ∗array, int size, bool own=false)

  Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (float ∗array, int size, bool own=false)

  Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (long long ∗array, int size, bool own=false)

  Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (int ∗array, int size, bool own=false)

  Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (short ∗array, int size, bool own=false)

  Creates a byte array object that wraps the given array.
- virtual ∼**ByteArrayAdapter** ()
- virtual int **getCapacity** () **const**

  Gets the size of the underlying array.
- virtual int **getCharCapacity** () **const**

Gets the size of the underlying array as if it contains chars.

- virtual int **getDoubleCapacity** () **const**

  Gets the size of the underlying array as if it contains doubles.
- virtual int **getFloatCapacity** () **const**

  Gets the size of the underlying array as if it contains doubles.
- virtual int **getLongCapacity** () **const**

  Gets the size of the underlying array as if it contains doubles.
- virtual int **getIntCapacity** () **const**

  Gets the size of the underlying array as if it contains ints.
- virtual int **getShortCapacity** () **const**

  Gets the size 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, int size, int offset, int length) **const**

  Reads from the Byte array starting at the specified offset and reading the specified length.
- virtual void **write** (unsigned char *buffer, int size, int offset, int length)

  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** (int size)

  Resizes the underlying array to the new given size, 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** () throw ()

  Clear all data from that Array, setting the underlying bytes to zero.
- unsigned char & **operator[]** (int index)

  *Allows the **ByteArrayAdapter*** (p. *484)* *to be indexed as a standard array.*
- **const** unsigned char & **operator[]** (int index) **const**
- virtual unsigned char **get** (int index) **const**

  Absolute get method.
- virtual char **getChar** (int index) **const**

  Reads one byte at the given index and returns it.
- virtual double **getDouble** (int index) **const**

  Reads eight bytes at the given index and returns it.
- virtual double **getDoubleAt** (int index) **const**

  Reads eight bytes at the given byte index and returns it.
- virtual float **getFloat** (int index) **const**

  Reads four bytes at the given index and returns it.
- virtual float **getFloatAt** (int index) **const**

  Reads four bytes at the given byte index and returns it.
- virtual long long **getLong** (int index) **const**

Reads eight bytes at the given index and returns it.

- virtual long long **getLongAt** (int index) **const**

  Reads eight bytes at the given byte index and returns it.

- virtual int **getInt** (int index) **const**

  Reads four bytes at the given index and returns it.

- virtual int **getIntAt** (int index) **const**

  Reads four bytes at the given byte index and returns it.

- virtual short **getShort** (int index) **const**

  Reads two bytes at the given index and returns it.

- virtual short **getShortAt** (int index) **const**

  Reads two bytes at the given byte index and returns it.

- virtual **ByteArrayAdapter** & **put** (int index, unsigned char value)

  Writes the given byte into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putChar** (int index, char value)

  Writes one byte containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putDouble** (int index, double value)

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putDoubleAt** (int index, double value)

  Writes eight bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putFloat** (int index, float value)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putFloatAt** (int index, float value)

  Writes four bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putLong** (int index, long long value)

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putLongAt** (int index, long long value)

  Writes eight bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putInt** (int index, int value)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putIntAt** (int index, int value)

  Writes four bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putShort** (int index, short value)

  Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putShortAt** (int index, short value)

  Writes two bytes containing the given value, into this buffer at the given byte index.

6.93.1 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.

Since

1.0

6.93.2 Constructor & Destructor Documentation

6.93.2.1 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int *size*)

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

size	The size of the array, this is the limit we read and write to.

Exceptions

IllegalArgumentException	if size is negative.

6.93.2.2 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (unsigned char * *array*, int *size*, bool *own* = `false`)

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	The physical array to wrap.
size	The size of the array, this is the limit we read and write to.
own	Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if the size is negative.

6.93.2.3 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (char * *array*, int *size*, bool *own* = `false`)

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	The physical array to wrap.
size	The size of the array, this is the limit we read and write to.
own	Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if the size is negative.

6.93.2.4 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (double * *array*, int *size*, bool *own* = false)

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	The physical array to wrap.
size	The size of the array, this is the limit we read and write to.
own	Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if the size is negative.

6.93.2.5 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (float * *array*, int *size*, bool *own* = false)

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	The physical array to wrap.
size	The size of the array, this is the limit we read and write to.
own	Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if the size is negative.

6.93.2.6 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (long long * *array*, int *size*, bool *own* = false)

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	The physical array to wrap.
size	The size of the array, this is the limit we read and write to.
own	Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if the size is negative.

6.93.2.7 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int * *array,* int *size,* bool *own* = false)

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	The physical array to wrap.
size	The size of the array, this is the limit we read and write to.
own	Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if the size is negative.

6.93.2.8 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (short * *array,* int *size,* bool *own* = false)

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	The physical array to wrap.
size	The size of the array, this is the limit we read and write to.
own	Indicates if this class is now the owner of the pointer.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if the size is negative.

6.93.2.9 virtual decaf::internal::util::ByteArrayAdapter::~ByteArrayAdapter ( ) [virtual]

6.93.3 Member Function Documentation

6.93.3.1 virtual void decaf::internal::util::ByteArrayAdapter::clear ( ) throw () [virtual]

Clear all data from that Array, setting the underlying bytes to zero.

6.93.3.2 virtual unsigned char decaf::internal::util::ByteArrayAdapter::get (int *index*) const [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

IndexOutOfBounds-Exception	If index is not smaller than the buffer's limit or is negative.

6.93.3.3 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. 484) objects that point to this array.

Returns

an unsigned char* pointer to the array this object wraps.

6.93.3.4 virtual int decaf::internal::util::ByteArrayAdapter::getCapacity ( ) const [inline, virtual]

Gets the size of the underlying array.

Returns

the size the array.

6.93.3.5 virtual char decaf::internal::util::ByteArrayAdapter::getChar (int *index*) const [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 byte that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	If index is not smaller than the buffer's limit or is negative.

6.93.3.6 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. 484) objects that point to this array.

Returns

an char* pointer to the array this object wraps.

6.93.3.7 virtual int decaf::internal::util::ByteArrayAdapter::getCharCapacity ( ) const [inline, virtual]

Gets the size of the underlying array as if it contains chars.

Returns

the size the array.

6.93.3.8 virtual double decaf::internal::util::ByteArrayAdapter::getDouble (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.9 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. 484) objects that point to this array.

Returns

an double∗ pointer to the array this object wraps.

6.93.3.10 virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.11 virtual int decaf::internal::util::ByteArrayAdapter::getDoubleCapacity ( ) const [inline, virtual]

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.93.3.12 virtual float decaf::internal::util::ByteArrayAdapter::getFloat (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.13 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. 484) objects that point to this array.

Returns

an float∗ pointer to the array this object wraps.

6.93.3.14 virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.15 virtual int decaf::internal::util::ByteArrayAdapter::getFloatCapacity ( ) const [inline, virtual]

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.93.3.16 virtual int decaf::internal::util::ByteArrayAdapter::getInt (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.17 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. 484) objects that point to this array.

Returns

an int∗ pointer to the array this object wraps.

6.93.3.18 virtual int decaf::internal::util::ByteArrayAdapter::getIntAt (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.19 virtual int decaf::internal::util::ByteArrayAdapter::getIntCapacity ( ) const [inline, virtual]

Gets the size of the underlying array as if it contains ints.

Returns

the size the array.

6.93.3.20 virtual long long decaf::internal::util::ByteArrayAdapter::getLong (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.21 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. 484) objects that point to this array.

Returns

an long long∗ pointer to the array this object wraps.

6.93.3.22 virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.23 virtual int decaf::internal::util::ByteArrayAdapter::getLongCapacity ( ) const [inline, virtual]

Gets the size of the underlying array as if it contains doubles.

Returns

the size the array.

6.93.3.24 virtual short decaf::internal::util::ByteArrayAdapter::getShort (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.25 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. 484) objects that point to this array.

Returns

an short∗ pointer to the array this object wraps.

6.93.3.26 virtual short decaf::internal::util::ByteArrayAdapter::getShortAt (int *index*) const [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 value at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

6.93.3.27 virtual int decaf::internal::util::ByteArrayAdapter::getShortCapacity ( ) const [inline, virtual]

Gets the size of the underlying array as if it contains shorts.

Returns

the size the array.

6.93.3.28 unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (int *index*)

Allows the **ByteArrayAdapter** (p. 484) to be indexed as a standard array.

calling the non constant version allows the user to change the value at index

Parameters

index	The position in the array to access, if the value is negative or greater than the size of the underlying array an IndexOutOfBoundsException is thrown.

Exceptions

IndexOutOfBounds-Exception	if the preconditions of index are not met.

6.93.3.29 const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (int *index*) const

6.93.3.30 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put (int *index,* unsigned char *value*) [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	The position in the Buffer to write the data.
value	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.31 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putChar (int *index,* char *value*) [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	The position in the Buffer to write the data.
value	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.32 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDouble (int *index,* double *value*) [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	The position in the Buffer to write the data.
value	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.33 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDoubleAt (int *index,* double *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

index	The position in the Buffer to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.34 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat (int *index,* float *value*) [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	The position in the Buffer to write the data.
value	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.35 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt (int *index,* float *value*) [virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

index	The position in the Buffer to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.36 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt (int *index,* int *value*) [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	The position in the Buffer to write the data.
value	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.37 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt (int *index,* int *value*) [virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

index	The position in the Buffer to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.38 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong (int *index,* long long *value*) [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	The position in the Buffer to write the data.
value	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.39 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt (int *index,* long long *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

index	The position in the Buffer to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.40 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShort (int *index,* short *value*) [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	The position in the Buffer to write the data.
value	The value to write to the array.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.41 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShortAt (int *index,* short *value*) [virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

Parameters

index	The position in the Buffer to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

6.93.3.42 virtual void decaf::internal::util::ByteArrayAdapter::read (unsigned char * *buffer,* int *size,* int *offset,* int *length*) const [virtual]

Reads from the Byte array starting at the specified offset and reading the specified length.

If the length is greater than the size of this underlying byte array then an BufferUnderflowException is thrown.

Parameters

buffer	The buffer to read data from this array into.
size	The size of the buffer passed.
offset	The position in this array to start reading from.
length	The amount of data to read from this array.

Exceptions

IndexOutOfBounds-Exception	if the offset + length exceeds the size.
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.

6.93.3.43 virtual void decaf::internal::util::ByteArrayAdapter::resize (int *size*) [virtual]

Resizes the underlying array to the new given size, 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. 484) can only be resized when it owns the underlying array, if it does not then it will throw an InvalidStateException.

Parameters

size	The new size of the array.

Exceptions

IllegalArgumentException	if the size parameter is negative.
InvalidStateException	if this object does not own the buffer.

6.93.3.44 virtual void decaf::internal::util::ByteArrayAdapter::write (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [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 size of this underlying byte array then an BufferOverflowException is thrown.

Parameters

buffer	The buffer to read data from this array into.
size	The size of the buffer passed.
offset	The position in this array to start reading from.
length	The amount of data to read from this array.

Exceptions

IndexOutOfBounds-Exception	if the offset + length exceeds the size.
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 size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ByteArrayAdapter.h**

6.94 decaf::internal::nio::ByteArrayBuffer Class Reference

This class defines six categories of operations upon byte buffers:

`#include <src/main/decaf/internal/nio/ByteArrayBuffer.h>`

Inheritance diagram for decaf::internal::nio::ByteArrayBuffer:

Public Member Functions

- **ByteArrayBuffer** (int **capacity**, bool readOnly=false)

  *Creates a **ByteArrayBuffer** (p. 502) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ByteArrayBuffer** (unsigned char ∗**array**, int size, int offset, int length, bool readOnly=false)

  *Creates a **ByteArrayBuffer** (p. 502) object that wraps the given array.*
- **ByteArrayBuffer** (**const decaf::lang::Pointer**< **ByteArrayAdapter** > &**array**, int offset, int length, bool readOnly=false)

  Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **ByteArrayBuffer** (**const ByteArrayBuffer** &other)

  *Create a **ByteArrayBuffer** (p. 502) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.*
- virtual ∼**ByteArrayBuffer** ()
- virtual bool **isReadOnly** () **const**

  Tells whether or not this buffer is read-only.

  Returns

  true if, and only if, this buffer is read-only
- virtual unsigned char ∗ **array** ()

  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. 1739)	*if this buffer is backed by an array but is read-only*
UnsupportedOperation-Exception	*if this buffer is not backed by an accessible array*

- virtual int **arrayOffset** () **const**

  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. 540).*
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. 1739)	*if this buffer is backed by an array but is read-only.*
UnsupportedOperation-Exception	*if this buffer is not backed by an accessible array.*

- virtual bool **hasArray** () **const**

  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. Subclasses 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.

- virtual **decaf::nio::CharBuffer** * **asCharBuffer** () **const**

  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. 451), which the caller then owns.*

- virtual **decaf::nio::DoubleBuffer** * **asDoubleBuffer** () **const**

  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. 451), which the caller then owns.*

- virtual **decaf::nio::FloatBuffer** * **asFloatBuffer** () **const**

  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. 451), which the caller then owns.*

- virtual **decaf::nio::IntBuffer** * **asIntBuffer** () **const**

  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. 451), which the caller then owns.*

- virtual **decaf::nio::LongBuffer** * **asLongBuffer** () **const**

  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. 451), which the caller then owns.*

- virtual **decaf::nio::ShortBuffer** * **asShortBuffer** () **const**

  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. 451), which the caller then owns.*

- virtual **ByteArrayBuffer** * **asReadOnlyBuffer** () **const**

  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.

- virtual **ByteArrayBuffer** & **compact** ()

  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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 535).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** * **duplicate** ()

  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. 451) which the caller owns.*

- virtual unsigned char **get** () **const**

  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. 474)	*if the buffer's current position is not smaller than its limit.*

- virtual unsigned char **get** (int index) **const**

  Absolute get method.
  Reads the byte at the given index.

Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual char **getChar** ()

  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. 474)	*if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual char **getChar** (int index) **const**

  Reads one byte at the given index and returns it.

  Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual double **getDouble** ()

  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. 474)	*if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual double **getDouble** (int index) **const**

  Reads eight bytes at the given index and returns it.

  Parameters

index	*The index in the **Buffer** (p. 451) where the bytes are to be read.*

  Returns

  the double at the given index in the buffer.

  Exceptions

IndexOutOfBoundsException	*if index is not smaller than the buffer's limit, or index is negative.*

- virtual float **getFloat** ()

  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. 474)	*if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual float **getFloat** (int index) **const**

  Reads four bytes at the given index and returns it.

Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual long long **getLong** ()

  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. 474)	*if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual long long **getLong** (int index) **const**

  Reads eight bytes at the given index and returns it.

  Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual int **getInt** ()

  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. 474)	*if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual int **getInt** (int index) **const**

  Reads four bytes at the given index and returns it.

  Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual short **getShort** ()

  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. 474)	*if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual short **getShort** (int index) **const**

  Reads two bytes at the given index and returns it.

Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual **ByteArrayBuffer** & **put** (unsigned char value)

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. 472)	*if this buffer's current position is not smaller than its limit.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **put** (int index, unsigned char value)

Writes the given byte into this buffer at the given index.

Parameters

index	*- position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **putChar** (char value)

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. 472)	*if there are fewer than bytes remaining in this buffer than the size of the data to be written*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **ByteArrayBuffer** & **putChar** (int index, char value)

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **ByteArrayBuffer** & **putDouble** (double value)

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. 472)	*if there are fewer than bytes remaining in this buffer than the size of the data to be written*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **ByteArrayBuffer** & **putDouble** (int index, double value)

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **putFloat** (float value)

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. 472)	*if there are fewer than bytes remaining in this buffer than the size of the data to be written.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **putFloat** (int index, float value)

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **putLong** (long long value)

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. 472)	*if there are fewer than bytes remaining in this buffer than the size of the data to be written.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **putLong** (int index, long long value)

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **putInt** (int value)

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. 472)	*if there are fewer than bytes remaining in this buffer than the size of the data to be written*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **ByteArrayBuffer** & **putInt** (int index, int value)

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **ByteArrayBuffer** & **putShort** (short value)

  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. 472)	*if there are fewer than bytes remaining in this buffer than the size of the data to be written.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** & **putShort** (int index, short value)

  Writes two bytes containing the given value, into this buffer at the given index.

  Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ByteArrayBuffer** ∗ **slice** () **const**

  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. 535) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 502) as Read-Only or not Read-Only.*

6.94.1 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. 518) float getFloat(int index) void **putFloat(float f)** (p. 523) void **putFloat(int index, float f)** (p. 524)

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

Since

1.0

6.94.2 Constructor & Destructor Documentation

6.94.2.1 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (int *capacity,* bool *readOnly* = `false`)

Creates a **ByteArrayBuffer** (p. 502) 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	The size of the array, this is the limit we read and write to.
readOnly	Should this buffer be read-only, default as false

Exceptions

IllegalArguementException	if the capacity value is negative.

6.94.2.2 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (unsigned char * *array*, int *size*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a **ByteArrayBuffer** (p. 502) object that wraps the given array.

Parameters

array	The array to wrap.
size	The size of the array passed.
offset	The position that is this buffers start position.
length	The size of the sub-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
IndexOutOfBounds-Exception	if the preconditions of size, offset and length are violated.

6.94.2.3 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **ByteArrayBuffer** (p. 502) will be that of the remaining capacity of the passed buffer.

Parameters

array	The ByteArrayAdapter to wrap
offset	The offset into array where the buffer starts
length	The length of the array we are wrapping or limit.
readOnly	Boolean indicating if this a readOnly buffer.

Exceptions

NullPointerException	if array is NULL
IndexOutOfBounds-Exception	if offset is greater than array capacity.

6.94.2.4 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (const ByteArrayBuffer & *other*)

Create a **ByteArrayBuffer** (p. 502) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.

Parameters

other	The **ByteArrayBuffer** (p. 502) this one is to mirror.

6.94.2.5 virtual decaf::internal::nio::ByteArrayBuffer::~ByteArrayBuffer ( ) [virtual]

6.94.3 Member Function Documentation

6.94.3.1 virtual unsigned char* decaf::internal::nio::ByteArrayBuffer::array ( ) [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. 1739)*	if this buffer is backed by an array but is read-only
UnsupportedOperation-Exception	if this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p. 539).

6.94.3.2 virtual int decaf::internal::nio::ByteArrayBuffer::arrayOffset ( ) const [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. 540).

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. 1739)*	if this buffer is backed by an array but is read-only.
UnsupportedOperation-Exception	if this buffer is not backed by an accessible array.

Implements **decaf::nio::ByteBuffer** (p. 540).

6.94.3.3 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** (p. 451), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 540).

6.94.3.4 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** (p. 451), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 540).

6.94.3.5 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** (p. 451), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 540).

6.94.3.6 virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteArrayBuffer::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** (p. 451), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 541).

6.94.3.7 virtual decaf::nio::LongBuffer* decaf::internal::nio::ByteArrayBuffer::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** (p. 451), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 541).

6.94.3.8 virtual ByteArrayBuffer* decaf::internal::nio::ByteArrayBuffer::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. 541).

6.94.3.9 virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteArrayBuffer::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** (p. 451), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 542).

6.94.3.10 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 535).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 542).

6.94.3.11 virtual ByteArrayBuffer* decaf::internal::nio::ByteArrayBuffer::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** (p. 451) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 542).

6.94.3.12 virtual unsigned char decaf::internal::nio::ByteArrayBuffer::get ( ) const [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. 474)*	if the buffer's current position is not smaller than its limit.

Implements **decaf::nio::ByteBuffer** (p. 543).

6.94.3.13 virtual unsigned char decaf::internal::nio::ByteArrayBuffer::get (int *index*) const [virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the byte is to be read.

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 544).

6.94.3.14 virtual char decaf::internal::nio::ByteArrayBuffer::getChar ( ) [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 *(p. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 544).

6.94.3.15 virtual char decaf::internal::nio::ByteArrayBuffer::getChar (int *index*) const [inline, virtual]

Reads one byte at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the byte is to be read.

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 544).

6.94.3.16 virtual double decaf::internal::nio::ByteArrayBuffer::getDouble ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 545).

6.94.3.17 virtual double decaf::internal::nio::ByteArrayBuffer::getDouble (int *index*) const [virtual]

Reads eight bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the double at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 545).

6.94.3.18 virtual float decaf::internal::nio::ByteArrayBuffer::getFloat ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 545).

6.94.3.19 virtual float decaf::internal::nio::ByteArrayBuffer::getFloat (int *index*) const [virtual]

Reads four bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the float at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 546).

6.94.3.20 virtual int decaf::internal::nio::ByteArrayBuffer::getInt ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 546).

6.94.3.21 virtual int decaf::internal::nio::ByteArrayBuffer::getInt (int *index*) const [virtual]

Reads four bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) 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, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 546).

6.94.3.22 virtual long long decaf::internal::nio::ByteArrayBuffer::getLong ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 547).

6.94.3.23 virtual long long decaf::internal::nio::ByteArrayBuffer::getLong (int *index*) const [virtual]

Reads eight bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the long long at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 547).

6.94.3.24 virtual short decaf::internal::nio::ByteArrayBuffer::getShort ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 547).

6.94.3.25 virtual short decaf::internal::nio::ByteArrayBuffer::getShort (int *index*) const [virtual]

Reads two bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the short at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 547).

6.94.3.26 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. Subclasses 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. 548).

6.94.3.27 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. 548).

6.94.3.28 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put (unsigned char *value*) [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. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 550).

6.94.3.29 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put (int *index,* unsigned char *value*) [virtual]

Writes the given byte into this buffer at the given index.

Parameters

index	- position in the **Buffer** (p. 451) to write the data
value	- the byte to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 550).

6.94.3.30 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putChar (char *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 550).

6.94.3.31 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putChar (int *index,* char *value*) [virtual]

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 551).

6.94.3.32 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putDouble (double *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 551).

6.94.3.33 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putDouble (int *index,* double *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 552).

6.94.3.34 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putFloat (float *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 552).

6.94.3.35 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putFloat (int *index,* float *value*) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 552).

6.94.3.36 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putInt (int *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 553).

6.94.3.37 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putInt (int *index,* int *value*) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 553).

6.94.3.38 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putLong (long long *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 554).

6.94.3.39 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putLong (int *index,* long long *value*) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 554).

6.94.3.40 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putShort (short *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 555).

6.94.3.41 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putShort (int *index,* short *value*) [virtual]

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 555).

6.94.3.42 virtual void decaf::internal::nio::ByteArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 502) as Read-Only or not Read-Only.

Parameters

value	Boolean value, true if this buffer is to be read-only, false otherwise.

6.94.3.43 virtual ByteArrayBuffer* decaf::internal::nio::ByteArrayBuffer::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. 535) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 555).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ByteArrayBuffer.h**

6.95 decaf::io::ByteArrayInputStream Class Reference

A **ByteArrayInputStream** (p. 527) contains an internal buffer that contains bytes that may be read from the stream.

`#include <src/main/decaf/io/ByteArrayInputStream.h>`

Inheritance diagram for decaf::io::ByteArrayInputStream:

Public Member Functions

- **ByteArrayInputStream** ()

  *Creates an **ByteArrayInputStream** (p. 527) with an empty input buffer, the buffer can be initialized with a call to setByteArray.*
- **ByteArrayInputStream** (**const** std::vector< unsigned char > &buffer)

  Creates the input stream and calls setBuffer with the specified buffer object.
- **ByteArrayInputStream** (**const** unsigned char *buffer, int bufferSize, bool own=false)

*Create an instance of the **ByteArrayInputStream** (p. 527) with the given buffer as the source of input for all read operations.*

- **ByteArrayInputStream** (**const** unsigned char *buffer, int bufferSize, int offset, int length, bool own=false)

  *Create an instance of the **ByteArrayInputStream** (p. 527) with the given buffer as the source of input for all read operations.*
- virtual ~**ByteArrayInputStream** ()
- virtual void **setByteArray** (**const** std::vector< unsigned char > &buffer)

  Sets the internal buffer.
- virtual void **setByteArray** (**const** unsigned char *buffer, int bufferSize)

  Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual void **setByteArray** (**const** unsigned char *buffer, int bufferSize, int offset, int length)

  Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual long long **skip** (long long num)

  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.
  *The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)	*if an I/O error occurs.*
UnsupportedOperation-Exception	*if the concrete stream class does not support skipping bytes.*

- 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.
  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.
  Calling mark on a closed stream instance should have no effect.
  The default implementation of this method does nothing.

  Parameters

readLimit	*The max bytes read before marked position is invalid.*

- virtual void **reset** ()

  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. 1198) might be thrown.*

– *If such an **IOException** (p. 1198) 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. 1198).*

– *If an **IOException** (p. 1198) 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.*

*The default implementation of this method throws an **IOException** (p. 1198).*

Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual bool **markSupported** () **const**

  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.
  The default implementation of this method returns false.

  Returns

  true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char ∗buffer, int size, int offset, int length)

6.95.1 Detailed Description

A **ByteArrayInputStream** (p. 527) contains an internal buffer that contains bytes that may be read from the stream.

An internal counter keeps track of the next byte to be supplied by the read method. The **ByteArrayInputStream** (p. 527) never copies the supplied buffers, only points to them, therefore the caller must ensure that the supplied buffer remain in scope, or is not deleted before this **ByteArrayInputStream** (p. 527) is freed. If the own argument of one of the constructors that accepts an array pointer is set to true than the **ByteArrayInputStream** (p. 527) instance will take ownership of the supplied pointer and delete it when that instance is destroyed.

Closing a **ByteArrayInputStream** (p. 527) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p. 1198).

Since

1.0

6.95.2 Constructor & Destructor Documentation

6.95.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( )

Creates an **ByteArrayInputStream** (p. 527) with an empty input buffer, the buffer can be initialized with a call to setByteArray.

6.95.2.2 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.

6.95.2.3 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * *buffer,* int *bufferSize,* bool *own* = false)

Create an instance of the **ByteArrayInputStream** (p. 527) with the given buffer as the source of input for all read operations.

Parameters

buffer	The initial byte array to use to read from.
bufferSize	The size of the buffer.
own	Indicates if this object should take ownership of the array, default is false.

Exceptions

NullPointerException	if the buffer is Null.
IllegalArguementException	if the bufferSize is negative.

6.95.2.4 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * *buffer,* int *bufferSize,* int *offset,* int *length,* bool *own* = false)

Create an instance of the **ByteArrayInputStream** (p. 527) with the given buffer as the source of input for all read operations.

Parameters

buffer	The initial byte array to use to read from.
bufferSize	The size of the buffer.
offset	The offset into the buffer to begin reading from.
length	The number of bytes to read past the offset.
own	Indicates if this object should take ownership of the array, default is false.

Exceptions

NullPointerException	if the buffer is Null.
IllegalArguementException	if the bufferSize is negative.

6.95.2.5 virtual decaf::io::ByteArrayInputStream::~ByteArrayInputStream ( ) [virtual]

6.95.3 Member Function Documentation

6.95.3.1 virtual int decaf::io::ByteArrayInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.95.3.2 virtual int decaf::io::ByteArrayInputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.95.3.3 virtual int decaf::io::ByteArrayInputStream::doReadByte ( ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1137).

6.95.3.4 virtual void decaf::io::ByteArrayInputStream::mark (int *readLimit*) [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.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::InputStream** (p. 1137).

6.95.3.5 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.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 1137).

6.95.3.6 virtual void decaf::io::ByteArrayInputStream::reset ( ) [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. 1198) might be thrown.
- If such an **IOException** (p. 1198) 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. 1198).
- If an **IOException** (p. 1198) 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.

The default implementation of this method throws an **IOException** (p. 1198).

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1140).

6.95.3.7 virtual void decaf::io::ByteArrayInputStream::setByteArray (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.

6.95.3.8 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * *buffer*, int *bufferSize*) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters

buffer	The initial byte array to use to read from.
bufferSize	The size of the buffer.

Exceptions

NullPointerException	if the buffer is Null.
IllegalArguementException	if the bufferSize is negative.

6.95.3.9 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * *buffer*, int *bufferSize*, int *offset*, int *length*) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters

buffer	The initial byte array to use to read from.
bufferSize	The size of the buffer.

offset	The offset into the buffer to begin reading from.
length	The number of bytes to read past the offset.

Exceptions

NullPointerException	if the buffer is Null.
IllegalArguementException	if the bufferSize is negative.

6.95.3.10 virtual long long decaf::io::ByteArrayInputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1140).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**ByteArrayInputStream.h**

6.96 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 of 32 bytes, the size increases as the need for more room arises.
- **ByteArrayOutputStream** (int bufferSize)

  *Creates a **ByteArrayOutputStream** (p. 533) with an internal buffer allocated with the given size.*

- virtual ~**ByteArrayOutputStream** ()
- std::pair< unsigned char *, int > **toByteArray** () **const**

  Creates a newly allocated byte array.
- long long **size** () **const**

  Gets the current count of bytes written into this ***ByteArrayOutputStream*** *(*p. *533).*
- virtual void **reset** ()

  Clear current Stream contents.
- virtual std::string **toString** () **const**

  Converts the bytes in the buffer into a standard C++ string.
- void **writeTo** (**OutputStream** *out) **const**

  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).

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (**const** unsigned char *buffer, int **size**, int offset, int length)

6.96.1 Constructor & Destructor Documentation

6.96.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ( )

Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.

6.96.1.2 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream (int *bufferSize*)

Creates a **ByteArrayOutputStream** (p. 533) with an internal buffer allocated with the given size.

Parameters

bufferSize	The size to use for the internal buffer.

Exceptions

IllegalArgumentException	if the size is less than or equal to zero.

6.96.1.3 virtual decaf::io::ByteArrayOutputStream::~ByteArrayOutputStream ( ) [virtual]

6.96.2 Member Function Documentation

6.96.2.1 virtual void decaf::io::ByteArrayOutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.96.2.2 virtual void decaf::io::ByteArrayOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 1602).

6.96.2.3 virtual void decaf::io::ByteArrayOutputStream::reset ( ) [virtual]

Clear current Stream contents.

Exceptions

IOException *(p. 1198)*	

6.96.2.4 long long decaf::io::ByteArrayOutputStream::size ( ) const

Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 533).

Returns

the number of valid bytes contained in the **ByteArrayOutputStream** (p. 533).

6.96.2.5 std::pair<unsigned char*, int> decaf::io::ByteArrayOutputStream::toByteArray ( ) const

Creates a newly allocated byte array.

Its size is the current size of this output stream and the valid contents of the buffer have been copied into it. The newly allocated array and its size are returned inside an STL pair structure, the caller is responsible for freeing the returned array.

Returns

an STL pair containing the copied array and its size.

6.96.2.6 virtual std::string decaf::io::ByteArrayOutputStream::toString ( ) const [virtual]

Converts the bytes in the buffer into a standard C++ string.

Returns

a string containing the bytes in the buffer

Reimplemented from **decaf::io::OutputStream** (p. 1603).

6.96.2.7 void decaf::io::ByteArrayOutputStream::writeTo (OutputStream * *out*) const

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**

6.97 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)

  Relative bulk get method.
- **ByteBuffer** & **get** (unsigned char *buffer, int size, int offset, int length)

  Relative bulk get method.
- **ByteBuffer** & **put** (**ByteBuffer** &src)

  This method transfers the bytes remaining in the given source buffer into this buffer.
- **ByteBuffer** & **put** (**const** unsigned char *buffer, int size, int offset, int length)

  This method transfers bytes into this buffer from the given source array.
- **ByteBuffer** & **put** (std::vector< unsigned char > &buffer)

  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

  Returns the byte array that backs this buffer.
- virtual int **arrayOffset** () **const** =0

  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

  Compacts this buffer.
- virtual **ByteBuffer** * **duplicate** ()=0

  Creates a new byte buffer that shares this buffer's content.
- virtual unsigned char **get** () **const** =0

  Relative get method.
- virtual unsigned char **get** (int index) **const** =0

  Absolute get method.
- virtual char **getChar** ()=0

  Reads the next byte at this buffer's current position, and then increments the position by one.
- virtual char **getChar** (int index) **const** =0

  Reads one byte at the given index and returns it.
- virtual double **getDouble** ()=0

  Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual double **getDouble** (int index) **const** =0

Reads eight bytes at the given index and returns it.

- virtual float **getFloat** ()=0

  Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual float **getFloat** (int index) **const** =0

  Reads four bytes at the given index and returns it.

- virtual long long **getLong** ()=0

  Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

- virtual long long **getLong** (int index) **const** =0

  Reads eight bytes at the given index and returns it.

- virtual int **getInt** ()=0

  Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual int **getInt** (int index) **const** =0

  Reads four bytes at the given index and returns it.

- virtual short **getShort** ()=0

  Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

- virtual short **getShort** (int index) **const** =0

  Reads two bytes at the given index and returns it.

- virtual **ByteBuffer** & **put** (unsigned char value)=0

  Writes the given byte into this buffer at the current position, and then increments the position.

- virtual **ByteBuffer** & **put** (int index, unsigned char value)=0

  Writes the given byte into this buffer at the given index.

- virtual **ByteBuffer** & **putChar** (char value)=0

  Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

- virtual **ByteBuffer** & **putChar** (int index, char value)=0

  Writes one byte containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putDouble** (double value)=0

  Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putDouble** (int index, double value)=0

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putFloat** (float value)=0

  Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putFloat** (int index, float value)=0

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putLong** (long long value)=0

  Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putLong** (int index, long long value)=0

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putInt** (int value)=0

  Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putInt** (int index, int value)=0

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putShort** (short value)=0

  Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putShort** (int index, short value)=0

  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**
- virtual bool **equals** (**const ByteBuffer** &value) **const**
- virtual bool **operator==** (**const ByteBuffer** &value) **const**
- virtual bool **operator<** (**const ByteBuffer** &value) **const**

Static Public Member Functions

- static **ByteBuffer** * **allocate** (int **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__, int size, int offset, int length)

  *Wraps the passed buffer with a new **ByteBuffer*** (p. *535).*
- static **ByteBuffer** * **wrap** (std::vector< unsigned char > &buffer)

  *Wraps the passed STL Byte Vector in a **ByteBuffer*** (p. *535).*

Protected Member Functions

- **ByteBuffer** (int **capacity**)

  *Creates a **ByteBuffer*** (p. *535) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.97.1 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. 545) float getFloat(int index) void **putFloat(float f)** (p. 552) void **putFloat(int index, float f)** (p. 552)

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. 1058) 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

6.97.2 Constructor & Destructor Documentation

6.97.2.1 decaf::nio::ByteBuffer::ByteBuffer (int *capacity*) [protected]

Creates a **ByteBuffer** (p. 535) 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	The size of the array, this is the limit we read and write to.

Exceptions

IllegalArgumentException	if capacity is negative.

6.97.2.2 virtual decaf::nio::ByteBuffer::~ByteBuffer ( ) [inline, virtual]

6.97.3 Member Function Documentation

6.97.3.1 static ByteBuffer* decaf::nio::ByteBuffer::allocate (int *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. 535) which the caller owns.

Exceptions

IllegalArgumentException	if capacity is negative.

6.97.3.2 virtual unsigned char* decaf::nio::ByteBuffer::array ( ) [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. 1739)*	if this buffer is backed by an array but is read-only
UnsupportedOperation-Exception	if this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 513).

6.97.3.3 virtual int decaf::nio::ByteBuffer::arrayOffset ( ) const [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. 540).

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. 1739)*	if this buffer is backed by an array but is read-only.
UnsupportedOperation-Exception	if this buffer is not backed by an accessible array.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 513).

6.97.3.4 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. 451), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 513).

6.97.3.5 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. 451), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 514).

6.97.3.6 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. 451), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 514).

6.97.3.7 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. 451), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 514).

6.97.3.8 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. 451), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 514).

6.97.3.9 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::ByteArrayBuffer** (p. 515).

6.97.3.10 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. 451), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 515).

6.97.3.11 virtual ByteBuffer& decaf::nio::ByteBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 535).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 515).

6.97.3.12 virtual int decaf::nio::ByteBuffer::compareTo (const ByteBuffer & *value*) const [virtual]

6.97.3.13 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. 451) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 516).

6.97.3.14 virtual bool decaf::nio::ByteBuffer::equals (const ByteBuffer & *value*) const [virtual]

6.97.3.15 ByteBuffer& decaf::nio::ByteBuffer::get (std::vector< unsigned char > *buffer*)

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. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length bytes remaining in this buffer

6.97.3.16 ByteBuffer& decaf::nio::ByteBuffer::get (unsigned char * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no bytes are transferred and a **BufferUnderflowException** (p. 474) 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	The pointer to an allocated buffer to fill.
size	The size of the passed in **Buffer** (p. 451).
offset	The position in the buffer to start filling.
length	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 451).

Exceptions

IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.
BufferUnderflowException *(p. 474)*	if there are fewer than length bytes remaining in this buffer.
NullPointerException	if the passed buffer is null.

6.97.3.17 virtual unsigned char decaf::nio::ByteBuffer::get ( ) const [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. 474)*	if the buffer's current position is not smaller than its limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 516).

6.97.3.18 virtual unsigned char decaf::nio::ByteBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the byte is to be read.

Returns

the byte that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 516).

6.97.3.19 virtual char decaf::nio::ByteBuffer::getChar ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 517).

6.97.3.20 virtual char decaf::nio::ByteBuffer::getChar (int *index*) const [pure virtual]

Reads one byte at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the byte is to be read.

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 517).

6.97.3.21 virtual double decaf::nio::ByteBuffer::getDouble ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 517).

6.97.3.22 virtual double decaf::nio::ByteBuffer::getDouble (int *index*) const [pure virtual]

Reads eight bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the double at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 517).

6.97.3.23 virtual float decaf::nio::ByteBuffer::getFloat ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 518).

6.97.3.24 virtual float decaf::nio::ByteBuffer::getFloat (int *index*) const [pure virtual]

Reads four bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the float at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 518).

6.97.3.25 virtual int decaf::nio::ByteBuffer::getInt ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 518).

6.97.3.26 virtual int decaf::nio::ByteBuffer::getInt (int *index*) const [pure virtual]

Reads four bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the int at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 519).

6.97.3.27 virtual long long decaf::nio::ByteBuffer::getLong ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 519).

6.97.3.28 virtual long long decaf::nio::ByteBuffer::getLong (int *index*) const [pure virtual]

Reads eight bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the long long at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 519).

6.97.3.29 virtual short decaf::nio::ByteBuffer::getShort ( ) [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. 474)*	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 520).

6.97.3.30 virtual short decaf::nio::ByteBuffer::getShort (int *index*) const [pure virtual]

Reads two bytes at the given index and returns it.

Parameters

index	The index in the **Buffer** (p. 451) where the bytes are to be read.

Returns

the short at the given index in the buffer.

Exceptions

IndexOutOfBounds-Exception	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 520).

6.97.3.31 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. Subclasses 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::ByteArrayBuffer** (p. 520).

6.97.3.32 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

Implements **decaf::nio::Buffer** (p. 454).

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 521).

6.97.3.33 virtual bool decaf::nio::ByteBuffer::operator< (const ByteBuffer & *value*) const [virtual]

6.97.3.34 virtual bool decaf::nio::ByteBuffer::operator== (const ByteBuffer & *value*) const [virtual]

6.97.3.35 ByteBuffer& decaf::nio::ByteBuffer::put (ByteBuffer & *src*)

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. 456), then no bytes are transferred and a **BufferOverflowException** (p. 472) 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. 472)*	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. 1739)*	if this buffer is read-only

6.97.3.36 ByteBuffer& decaf::nio::ByteBuffer::put (const unsigned char * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no bytes are transferred and a **BufferOverflowException** (p. 472) 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.
size	The size of the given array.
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. 472)*	if there is insufficient space in this buffer.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.97.3.37 ByteBuffer& decaf::nio::ByteBuffer::put (std::vector< unsigned char > & *buffer*)

This method transfers the entire content of the given source byte array into this buffer.

This is the same as calling put(&buffer[0], buffer.size(), 0, buffer.size())

Parameters

buffer	The buffer whose contents are copied to this **ByteBuffer** (p. 535).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

6.97.3.38 virtual ByteBuffer& decaf::nio::ByteBuffer::put (unsigned char *value*) [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. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 521).

6.97.3.39 virtual ByteBuffer& decaf::nio::ByteBuffer::put (int *index,* unsigned char *value*) [pure virtual]

Writes the given byte into this buffer at the given index.

Parameters

index	- position in the **Buffer** (p. 451) to write the data
value	- the byte to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 521).

6.97.3.40 virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (char *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 522).

6.97.3.41 virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (int *index,* char *value*) [pure virtual]

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 522).

6.97.3.42 virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (double *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 522).

6.97.3.43 virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (int *index*, double *value*) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 523).

6.97.3.44 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (float *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 523).

6.97.3.45 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (int *index*, float *value*) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 524).

6.97.3.46 virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 524).

6.97.3.47 virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int *index,* int *value*) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 524).

6.97.3.48 virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (long long *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 525).

6.97.3.49 virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (int *index,* long long *value*) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The value to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 525).

6.97.3.50 virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (short *value*) [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. 472)*	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 526).

6.97.3.51 virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (int *index,* short *value*) [pure virtual]

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 526).

6.97.3.52 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. 535) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 527).

6.97.3.53 virtual std::string decaf::nio::ByteBuffer::toString ( ) const [virtual]

Returns

a std::string describing this object

6.97.3.54 static ByteBuffer* decaf::nio::ByteBuffer::wrap (unsigned char * *array,* int *size,* int *offset,* int *length*) [static]

Wraps the passed buffer with a new **ByteBuffer** (p. 535).

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.
size	The size of the provided array.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new **ByteBuffer** (p. 535) that is backed by buffer, caller owns.

Exceptions

NullPointerException	if the array passed in is NULL.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.97.3.55 static ByteBuffer* decaf::nio::ByteBuffer::wrap (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 535).

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. 535) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ByteBuffer.h**

6.98 cms::BytesMessage Class Reference

A **BytesMessage** (p. 557) 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** () throw ()
- virtual void **setBodyBytes** (**const** unsigned char ∗buffer, int numBytes)=0

  sets the bytes given to the message body.
- virtual unsigned char ∗ **getBodyBytes** () **const** =0

  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 int **getBodyLength** () **const** =0

  Returns the number of bytes contained in the body of this message.
- virtual void **reset** ()=0

  Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () **const** =0

  Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value)=0

  Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () **const** =0

  Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value)=0

  Writes a byte to the bytes message stream as a 1-byte value.
- virtual int **readBytes** (std::vector< unsigned char > &value) **const** =0

  Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (**const** std::vector< unsigned char > &value)=0

  Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.
- virtual int **readBytes** (unsigned char ∗buffer, int length) **const** =0

  Reads a portion of the bytes message stream.
- virtual void **writeBytes** (**const** unsigned char ∗value, int offset, int length)=0

  Writes a portion of a byte array to the bytes message stream.
- virtual char **readChar** () **const** =0

  Reads a Char from the Bytes message stream.

- virtual void **writeChar** (char value)=0

  Writes a char to the bytes message stream as a 1-byte value.
- virtual float **readFloat** () **const** =0

  Reads a 32 bit float from the Bytes message stream.
- virtual void **writeFloat** (float value)=0

  Writes a float to the bytes message stream as a 4 byte value.
- virtual double **readDouble** () **const** =0

  Reads a 64 bit double from the Bytes message stream.
- virtual void **writeDouble** (double value)=0

  Writes a double to the bytes message stream as a 8 byte value.
- virtual short **readShort** () **const** =0

  Reads a 16 bit signed short from the Bytes message stream.
- virtual void **writeShort** (short value)=0

  Writes a signed short to the bytes message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () **const** =0

  Reads a 16 bit unsigned short from the Bytes message stream.
- virtual void **writeUnsignedShort** (unsigned short value)=0

  Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual int **readInt** () **const** =0

  Reads a 32 bit signed integer from the Bytes message stream.
- virtual void **writeInt** (int value)=0

  Writes a signed int to the bytes message stream as a 4 byte value.
- virtual long long **readLong** () **const** =0

  Reads a 64 bit long from the Bytes message stream.
- virtual void **writeLong** (long long value)=0

  Writes a long long to the bytes message stream as a 8 byte value.
- virtual std::string **readString** () **const** =0

  Reads an ASCII String from the Bytes message stream.
- virtual void **writeString** (**const** std::string &value)=0

  Writes an ASCII String to the Bytes message stream.
- virtual std::string **readUTF** () **const** =0

  *Reads an UTF String from the **BytesMessage** (p. 557) stream.*
- virtual void **writeUTF** (**const** std::string &value)=0

  *Writes an UTF String to the **BytesMessage** (p. 557) stream.*
- virtual **BytesMessage** * **clone** () **const** =0

  Clones this message.

6.98.1 Detailed Description

A **BytesMessage** (p. 557) object is used to send a message containing a stream of unsigned bytes.

It inherits from the **Message** (p. 1426) 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. 557) interface.

The **BytesMessage** (p. 557) methods are based largely on those found in **decaf.io.DataInputStream** (p. 849) and **decaf.io.DataOutputStream** (p. 860).

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. 1490) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 1490) is thrown.

Since

1.0

6.98.2 Constructor & Destructor Documentation

6.98.2.1 virtual cms::BytesMessage::~BytesMessage ( ) throw () [virtual]

6.98.3 Member Function Documentation

6.98.3.1 virtual BytesMessage* cms::BytesMessage::clone ( ) const [pure virtual]

Clones this message.

Returns

a deep copy of this message.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs while cloning the **Message** (p. 1426).

Implements **cms::Message** (p. 1430).

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 131).

6.98.3.2 virtual unsigned char* cms::BytesMessage::getBodyBytes ( ) const [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. 640)*	- If an internal error occurs.
MessageNotReadable-Exception *(p. 1490)*	- If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 132).

6.98.3.3 virtual int cms::BytesMessage::getBodyLength ( ) const [pure virtual]

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
MessageNotReadable-Exception *(p. 1490)*	- If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 132).

6.98.3.4 virtual bool cms::BytesMessage::readBoolean ( ) const [pure virtual]

Reads a Boolean from the Bytes message stream.

Returns

boolean value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 133).

6.98.3.5 virtual unsigned char cms::BytesMessage::readByte ( ) const [pure virtual]

Reads a Byte from the Bytes message stream.

Returns

unsigned char value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 133).

6.98.3.6 virtual int cms::BytesMessage::readBytes (std::vector< unsigned char > & *value*) const [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. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 133).

6.98.3.7 virtual int cms::BytesMessage::readBytes (unsigned char * *buffer*, int *length*) const [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

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 134).

6.98.3.8 virtual char cms::BytesMessage::readChar ( ) const [pure virtual]

Reads a Char from the Bytes message stream.

Returns

char value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 135).

6.98.3.9 virtual double cms::BytesMessage::readDouble ( ) const [pure virtual]

Reads a 64 bit double from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 135).

6.98.3.10 virtual float cms::BytesMessage::readFloat ( ) const [pure virtual]

Reads a 32 bit float from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 135).

6.98.3.11 virtual int cms::BytesMessage::readInt ( ) const [pure virtual]

Reads a 32 bit signed integer from the Bytes message stream.

Returns

int value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 135).

6.98.3.12 virtual long long cms::BytesMessage::readLong ( ) const [pure virtual]

Reads a 64 bit long from the Bytes message stream.

Returns

long long value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 136).

6.98.3.13 virtual short cms::BytesMessage::readShort ( ) const [pure virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns

short value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 136).

6.98.3.14 virtual std::string cms::BytesMessage::readString ( ) const [pure virtual]

Reads an ASCII String from the Bytes message stream.

Returns

String from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 136).

6.98.3.15 virtual unsigned short cms::BytesMessage::readUnsignedShort ( ) const [pure virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns

unsigned short value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 137).

6.98.3.16 virtual std::string cms::BytesMessage::readUTF ( ) const [pure virtual]

Reads an UTF String from the **BytesMessage** (p. 557) stream.

Returns

String from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of bytes stream has been reached.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 137).

6.98.3.17 virtual void cms::BytesMessage::reset ( ) [pure virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

CMSException *(p. 640)*	- If the provider fails to perform the reset operation.
MessageFormatException *(p. 1478)*	- If the **Message** (p. 1426) has an invalid format.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 137).

6.98.3.18 virtual void cms::BytesMessage::setBodyBytes (const unsigned char * *buffer,* int *numBytes*) [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. 640)*	- If an internal error occurs.
MessageNotWriteable-Exception *(p. 1490)*	- if in Read Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 138).

6.98.3.19 virtual void cms::BytesMessage::writeBoolean (bool *value*) [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

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 138).

6.98.3.20 virtual void cms::BytesMessage::writeByte (unsigned char *value*) [pure virtual]

Writes a byte to the bytes message stream as a 1-byte value.

Parameters

value	byte to write to the stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 138).

6.98.3.21 virtual void cms::BytesMessage::writeBytes (const std::vector< unsigned char > & *value*) [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

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 139).

6.98.3.22 virtual void cms::BytesMessage::writeBytes (const unsigned char * *value*, int *offset*, int *length*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 139).

6.98.3.23 virtual void cms::BytesMessage::writeChar (char *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 139).

6.98.3.24 virtual void cms::BytesMessage::writeDouble (double *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 140).

6.98.3.25 virtual void cms::BytesMessage::writeFloat (float *value*) [pure virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters

value	float to write to the stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 140).

6.98.3.26 virtual void cms::BytesMessage::writeInt (int *value*) [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

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 140).

6.98.3.27 virtual void cms::BytesMessage::writeLong (long long *value*) [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

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 141).

6.98.3.28 virtual void cms::BytesMessage::writeShort (short *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 141).

6.98.3.29 virtual void cms::BytesMessage::writeString (const std::string & *value*) [pure virtual]

Writes an ASCII String to the Bytes message stream.

Parameters

value	String to write to the stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 141).

6.98.3.30 virtual void cms::BytesMessage::writeUnsignedShort (unsigned short *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 141).

6.98.3.31 virtual void cms::BytesMessage::writeUTF (const std::string & *value*) [pure virtual]

Writes an UTF String to the **BytesMessage** (p. 557) stream.

Parameters

value	String to write to the stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 142).

The documentation for this class was generated from the following file:

- src/main/cms/**BytesMessage.h**

6.99 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** () throw ()
- virtual void **close** ()

  Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual void **start** ()

  Starts the service.
- virtual void **stop** ()

  Stops this service.
- virtual **cms::Message** ∗ **receive** ()

  Synchronously Receive a Message.
- virtual **cms::Message** ∗ **receive** (int millisecs)

  Synchronously Receive a Message, time out after defined interval.
- virtual **cms::Message** ∗ **receiveNoWait** ()

  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)

  Sets the MessageListener that this class will send notifs on.
- virtual **cms::MessageListener** ∗ **getMessageListener** () **const**

  Gets the MessageListener that this class will send mew Message notification events to.
- virtual std::string **getMessageSelector** () **const**

  Gets this message consumer's message selector expression.

6.99.1 Detailed Description

A cached message consumer contained within a pooled session.

6.99.2 Constructor & Destructor Documentation

6.99.2.1 activemq::cmsutil::CachedConsumer::CachedConsumer (cms::MessageConsumer ∗ *consumer*) [inline]

6.99.2.2 virtual activemq::cmsutil::CachedConsumer::∼CachedConsumer ( ) throw () [inline, virtual]

6.99.3 Member Function Documentation

6.99.3.1 virtual void activemq::cmsutil::CachedConsumer::close ( ) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 633).

6.99.3.2 virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener ( ) const [inline, virtual]

Gets the MessageListener that this class will send mew Message notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1456).

6.99.3.3 virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector ( ) const [inline, virtual]

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1456).

6.99.3.4 virtual cms::Message* activemq::cmsutil::CachedConsumer::receive ( ) [inline, virtual]

Synchronously Receive a Message.

Returns

new message which the caller owns and must delete.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1457).

6.99.3.5 virtual cms::Message* activemq::cmsutil::CachedConsumer::receive (int *millisecs*) [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

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1457).

6.99.3.6 virtual cms::Message* activemq::cmsutil::CachedConsumer::receiveNoWait ( ) [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

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1457).

6.99.3.7 virtual void activemq::cmsutil::CachedConsumer::setMessageListener (cms::MessageListener * *listener*) [inline, virtual]

Sets the MessageListener that this class will send notifs on.

Parameters

listener	The listener of messages received by this consumer.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1458).

6.99.3.8 virtual void activemq::cmsutil::CachedConsumer::start ( ) [inline, virtual]

Starts the service.

Exceptions

CMSException	if an internal error occurs while starting.

Implements **cms::Startable** (p. 1964).

6.99.3.9 virtual void activemq::cmsutil::CachedConsumer::stop ( ) [inline, virtual]

Stops this service.

Exceptions

CMSException	- if an internal error occurs while stopping the Service.

Implements **cms::Stoppable** (p. 2017).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedConsumer.h**

6.100 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** () throw ()
- virtual void **close** ()

  Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual void **send** (**cms::Message** ∗message)

  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)

  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)

  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)

  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)

  Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () **const**

  Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value)

  Sets if Message Ids are disabled for this Producer.
- virtual bool **getDisableMessageID** () **const**

  Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value)

  Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () **const**

Gets if Message Time Stamps are disabled for this Producer.

- virtual void **setPriority** (int priority)

  Sets the Priority that this Producers sends messages at.

- virtual int **getPriority** () **const**

  Gets the Priority level that this producer sends messages at.

- virtual void **setTimeToLive** (long long time)

  Sets the Time to Live that this Producers sends messages with.

- virtual long long **getTimeToLive** () **const**

  Gets the Time to Live that this producer sends messages with.

6.100.1 Detailed Description

A cached message producer contained within a pooled session.

6.100.2 Constructor & Destructor Documentation

6.100.2.1 activemq::cmsutil::CachedProducer::CachedProducer (cms::MessageProducer * *producer*) [inline]

6.100.2.2 virtual activemq::cmsutil::CachedProducer::~CachedProducer ( ) throw () [inline, virtual]

6.100.3 Member Function Documentation

6.100.3.1 virtual void activemq::cmsutil::CachedProducer::close ( ) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 633).

6.100.3.2 virtual int activemq::cmsutil::CachedProducer::getDeliveryMode ( ) const [inline, virtual]

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1493).

6.100.3.3 virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID ( ) const [inline, virtual]

Gets if Message Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1493).

6.100.3.4 virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp ( ) const [inline, virtual]

Gets if Message Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1493).

6.100.3.5 virtual int activemq::cmsutil::CachedProducer::getPriority ( ) const [inline, virtual]

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1494).

6.100.3.6 virtual long long activemq::cmsutil::CachedProducer::getTimeToLive ( ) const [inline, virtual]

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1494).

6.100.3.7 virtual void activemq::cmsutil::CachedProducer::send (cms::Message * *message*) [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

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.
UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1494).

6.100.3.8 virtual void activemq::cmsutil::CachedProducer::send (cms::Message * *message,* int *deliveryMode,* int *priority,* long long *timeToLive*) [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

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.
UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1495).

6.100.3.9 virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * *destination,* cms::Message * *message*) [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

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.

UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1495).

6.100.3.10 virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * *destination*, cms::Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) [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

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.
UnsupportedOperation-Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 1496).

6.100.3.11 virtual void activemq::cmsutil::CachedProducer::setDeliveryMode (int *mode*) [inline, virtual]

Sets the delivery mode for this Producer.

Parameters

mode	The DeliveryMode

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1496).

6.100.3.12 virtual void activemq::cmsutil::CachedProducer::setDisableMessageID (bool *value*) [inline, virtual]

Sets if Message Ids are disabled for this Producer.

Parameters

value	boolean indicating enable / disable (true / false)

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1496).

6.100.3.13 virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp (bool *value*) [inline, virtual]

Sets if Message Time Stamps are disabled for this Producer.

Parameters

value	- boolean indicating enable / disable (true / false)

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1497).

6.100.3.14 virtual void activemq::cmsutil::CachedProducer::setPriority (int *priority*) [inline, virtual]

Sets the Priority that this Producers sends messages at.

Parameters

priority	int value for Priority level

Exceptions

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1497).

6.100.3.15 virtual void activemq::cmsutil::CachedProducer::setTimeToLive (long long *time*) [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

CMSException	- if an internal error occurs.

Implements **cms::MessageProducer** (p. 1497).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedProducer.h**

6.101 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

  Computes a result, or throws an exception if unable to do so.

6.101.1 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. 578) object can return a result and is allowed to throw an exceptions from its call method.

The **Executors** (p. 1005) class contains utility methods to convert from other common forms to **Callable** (p. 578) classes.

Since

1.0

6.101.2 Constructor & Destructor Documentation

6.101.2.1 template<typename V > virtual decaf::util::concurrent::Callable< V >::~Callable ( ) [inline, virtual]

6.101.3 Member Function Documentation

6.101.3.1 template<typename V > virtual V decaf::util::concurrent::Callable< V >::call ( ) [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**

6.102 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.

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

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy:

Data Structures

- class **DiscardOldestPolicy**

  *Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always destroys the oldest unexecuted task in the **Queue** (p. 1723) and then attempts to execute the rejected task using the passed in executor.*
- class **DiscardPolicy**

  *Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always destroys the rejected task and returns quietly.*

Public Member Functions

- **CallerRunsPolicy** ()
- virtual ∼**CallerRunsPolicy** ()
- virtual void **rejectedExecution** (**decaf::lang::Runnable** ∗task, **ThreadPoolExecutor** ∗executer **DECAF_UNUSED**)

6.102.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.

Since

1.0

6.102.2 Constructor & Destructor Documentation

6.102.2.1 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::CallerRunsPolicy ( ) [inline]

6.102.2.2 virtual decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::∼CallerRunsPolicy ( ) [inline, virtual]

6.102.3 Member Function Documentation

6.102.3.1 virtual void decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::rejectedExecution (decaf::lang::Runnable ∗ *task*, ThreadPoolExecutor ∗executer *DECAF_UNUSED*) [inline, virtual]

References decaf::lang::Runnable::run().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.103 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 ()

6.103.1 Constructor & Destructor Documentation

6.103.1.1 decaf::util::concurrent::CancellationException::CancellationException ( ) throw () [inline]

Default Constructor.

6.103.1.2 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

6.103.1.3 decaf::util::concurrent::CancellationException::CancellationException (const CancellationException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	- The Exception to copy in this new instance.

6.103.1.4 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.

6.103.1.5 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

6.103.1.6 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

6.103.1.7 virtual decaf::util::concurrent::CancellationException::~CancellationException ( ) throw () [inline, virtual]

6.103.2 Member Function Documentation

6.103.2.1 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. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CancellationException.h**

6.104 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

  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

  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

  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.

6.104.1 Detailed Description

Base interface for all identity certificates.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 virtual decaf::security::cert::Certificate::~Certificate ( ) [inline, virtual]

6.104.3 Member Function Documentation

6.104.3.1 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.

6.104.3.2 virtual void decaf::security::cert::Certificate::getEncoded (std::vector< unsigned char > & *output*) const [pure virtual]

Provides the encoded form of this certificate.

Parameters

output	Receives the encoded form of this certificate.

Exceptions

CertificateEncoding-Exception *(p. 585)*	if an encoding error occurs

6.104.3.3 virtual PublicKey* decaf::security::cert::Certificate::getPublicKey ( ) [pure virtual]

Gets the public key of this certificate.

Returns

the public key

6.104.3.4 virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey ( ) const [pure virtual]

Gets the public key of this certificate.

Returns

the public key

6.104.3.5 virtual std::string decaf::security::cert::Certificate::getType ( ) const [pure virtual]

Returns the type of this certificate.

Returns

the type of this certificate

6.104.3.6 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

6.104.3.7 virtual void decaf::security::cert::Certificate::verify (const PublicKey & *publicKey*) const [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. 1535)*	- on unsupported signature algorithms.
InvalidKeyException *(p. 1191)*	- on incorrect key.
NoSuchProviderException *(p. 1539)*	- if there's no default provider.
SignatureException *(p. 1889)*	- on signature errors.
CertificateException *(p. 587)*	- on encoding errors.

6.104.3.8 virtual void decaf::security::cert::Certificate::verify (const PublicKey & *publicKey,* const std::string & *sigProvider*) const [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. 1535)*	- on unsupported signature algorithms.
InvalidKeyException *(p. 1191)*	- on incorrect key.
NoSuchProviderException *(p. 1539)*	- if there's no default provider.
SignatureException *(p. 1889)*	- on signature errors.
CertificateException *(p. 587)*	- on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**

6.105 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 occured.
- virtual
  CertificateEncodingException ∗ **clone** () **const**

  Clones this exception.
- virtual ∼**CertificateEncodingException** () throw ()

6.105.1 Constructor & Destructor Documentation

6.105.1.1 decaf::security::cert::CertificateEncodingException::CertificateEncodingException ( ) throw () [inline]

Default Constructor.

6.105.1.2 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

6.105.1.3 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const CertificateEncodingException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.105.1.4 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 occured.

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

6.105.1.5 virtual decaf::security::cert::CertificateEncodingException::~CertificateEncodingException ( ) throw () [inline, virtual]

6.105.2 Member Function Documentation

6.105.2.1 virtual CertificateEncodingException* decaf::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. 588).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateEncodingException.h**

6.106 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 occured.
- virtual **CertificateException** * **clone** () **const**

  Clones this exception.
- virtual ~**CertificateException** () throw ()

6.106.1 Constructor & Destructor Documentation

6.106.1.1 decaf::security::cert::CertificateException::CertificateException ( ) throw () [inline]

Default Constructor.

6.106.1.2 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

6.106.1.3 decaf::security::cert::CertificateException::CertificateException (const CertificateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.106.1.4 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 occured.

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

6.106.1.5 virtual decaf::security::cert::CertificateException::~CertificateException ( ) throw () [inline, virtual]

6.106.2 Member Function Documentation

6.106.2.1 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. 1081).

Reimplemented in **decaf::security::cert::CertificateExpiredException** (p. 590), **decaf::security::cert::CertificateNotYetValidException** (p. 591), **decaf::security::cert::CertificateParsingException** (p. 593), and **decaf::security::cert::CertificateEncodingException** (p. 587).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateException.h**

6.107 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 occured.
- virtual **CertificateExpiredException** * **clone** () **const**
  Clones this exception.
- virtual ~**CertificateExpiredException** () throw ()

6.107.1 Constructor & Destructor Documentation

6.107.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( ) throw () [inline]

Default Constructor.

6.107.1.2 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

6.107.1.3 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const CertificateExpiredException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.107.1.4 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 occured.

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

6.107.1.5 virtual decaf::security::cert::CertificateExpiredException::~CertificateExpiredException ( ) throw () [inline, virtual]

6.107.2 Member Function Documentation

6.107.2.1 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. 588).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateExpiredException.h**

6.108 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 occured.
- virtual
  CertificateNotYetValidException * **clone** () **const**

  Clones this exception.
- virtual ~**CertificateNotYetValidException** () throw ()

6.108.1 Constructor & Destructor Documentation

6.108.1.1 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException ( ) throw () [inline]

Default Constructor.

6.108.1.2 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

6.108.1.3 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const CertificateNotYetValidException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.108.1.4 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 occured.

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

6.108.1.5 virtual decaf::security::cert::CertificateNotYetValidException::~CertificateNotYetValidException ( ) throw () [inline, virtual]

6.108.2 Member Function Documentation

6.108.2.1 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. 588).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateNotYetValidException.h**

6.109 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 occured.
- virtual
  CertificateParsingException * **clone** () **const**

  Clones this exception.
- virtual ~**CertificateParsingException** () throw ()

6.109.1 Constructor & Destructor Documentation

6.109.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException ( ) throw () [inline]

Default Constructor.

6.109.1.2 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

6.109.1.3 decaf::security::cert::CertificateParsingException::CertificateParsingException (const CertificateParsingException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.109.1.4 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 occured.

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

6.109.1.5 virtual decaf::security::cert::CertificateParsingException::~CertificateParsingException ( ) throw () [inline, virtual]

6.109.2 Member Function Documentation

6.109.2.1 virtual CertificateParsingException* decaf::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. 588).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateParsingException.h**

6.110 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. 593) 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. 593) 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. 593) 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 charactor in bits.

6.110.1 Constructor & Destructor Documentation

6.110.1.1 decaf::lang::Character::Character (char *value*)

Parameters

value	- char to wrap.

6.110.2 Member Function Documentation

6.110.2.1 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.

Reimplemented from **decaf::lang::Number** (p. 1544).

6.110.2.2 virtual int decaf::lang::Character::compareTo (const Character & *c*) const [inline, virtual]

Compares this **Character** (p. 593) instance with another.

Parameters

c	- the **Character** (p. 593) 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 repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< Character >** (p. 687).

6.110.2.3 virtual int decaf::lang::Character::compareTo (const char & *c*) const [inline, virtual]

Compares this **Character** (p. 593) 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 repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< char >** (p. 687).

6.110.2.4 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 radix + 'A' - 10. In this case, ch - 'A' + 10 is returned.
  - The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than radix + 'a' - 10. In this case, 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

6.110.2.5 virtual double decaf::lang::Character::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.110.2.6 bool decaf::lang::Character::equals (const Character & *c*) const [inline, virtual]

Returns

true if the two **Character** (p. 593) Objects have the same value.

Implements **decaf::lang::Comparable**< **Character** > (p. 688).

6.110.2.7 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. 688).

6.110.2.8 virtual float decaf::lang::Character::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.110.2.9 virtual int decaf::lang::Character::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.110.2.10 static bool decaf::lang::Character::isDigit (char *c*) [inline, static]

Indicates whether or not the given character is a digit.

6.110.2.11 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

6.110.2.12 static bool decaf::lang::Character::isLetter (char *c*) [inline, static]

Indicates whether or not the given character is a letter.

6.110.2.13 static bool decaf::lang::Character::isLetterOrDigit (char *c*) [inline, static]

Indicates whether or not the given character is either a letter or a digit.

6.110.2.14 static bool decaf::lang::Character::isLowerCase (char *c*) [inline, static]

Indicates whether or not the given character is a lower case character.

6.110.2.15 static bool decaf::lang::Character::isUpperCase (char *c*) [inline, static]

Indicates whether or not the given character is a upper case character.

6.110.2.16 static bool decaf::lang::Character::isWhitespace (char *c*) [inline, static]

Indicates whether or not the given character is considered whitespace.

6.110.2.17 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.

Implements **decaf::lang::Number** (p. 1545).

6.110.2.18 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.

Implements **decaf::lang::Comparable**< **Character** > (p. 688).

6.110.2.19 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. 688).

6.110.2.20 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.

Implements **decaf::lang::Comparable**< **Character** > (p. 688).

6.110.2.21 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. 688).

6.110.2.22 virtual short decaf::lang::Character::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1545).

6.110.2.23 std::string decaf::lang::Character::toString ( ) const

Returns

this **Character** (p. 593) Object as a **String** (p. 2031) Representation

6.110.2.24 static Character decaf::lang::Character::valueOf (char *value*) [inline, static]

Returns a **Character** (p. 593) instance representing the specified char value.

Parameters

value	- the primitive char to wrap.

Returns

a new Charactor instance that wraps this value.

6.110.3 Field Documentation

6.110.3.1 const int decaf::lang::Character::MAX_RADIX = 36 [static]

The maximum radix available for conversion to and from strings.

6.110.3.2 const char decaf::lang::Character::MAX_VALUE = (char)0x80 [static]

The maximum value that a signed char can take on.

6.110.3.3 const int decaf::lang::Character::MIN_RADIX = 2 [static]

The minimum radix available for conversion to and from strings.

6.110.3.4 const char decaf::lang::Character::MIN_VALUE = (char)0x7F [static]

The minimum value that a signed char can take on.

6.110.3.5 const int decaf::lang::Character::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/**Character.h**

6.111 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** (int size, bool **readOnly**=false)

  *Creates a **CharArrayBuffer** (p. 600) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **CharArrayBuffer** (char ∗**array**, int size, int **offset**, int **length**, bool **readOnly**=false)

  *Creates a **CharArrayBuffer** (p. 600) object that wraps the given array.*
- **CharArrayBuffer** (**const decaf::lang::Pointer**< **ByteArrayAdapter** > &**array**, int **offset**, int **length**, bool **readOnly**=false)

  Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

- **CharArrayBuffer** (**const CharArrayBuffer** &other)

  *Create a **CharArrayBuffer** (p. 600) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ∼**CharArrayBuffer** ()
- virtual char ∗ **array** ()

  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. 451).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperationException	*if the underlying store has no array.*

- virtual int **arrayOffset** ()

  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. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperationException	*if the underlying store has no array.*

- virtual **CharBuffer** ∗ **asReadOnlyBuffer** () **const**

  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.

- virtual **CharBuffer** & **compact** ()

  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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 609).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*- If this buffer is read-only*

- virtual **CharBuffer** ∗ **duplicate** ()

  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. 451) which the caller owns.*

- virtual char **get** ()

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. 474)	*if there no more data to return*

- virtual char **get** (int index) **const**

Absolute get method.
Reads the char at the given index.

Parameters

index	*The index in the **Buffer** (p. 451) 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 or is negative.*

- virtual bool **hasArray** () **const**

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. Subclasses 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

- virtual bool **isReadOnly** () **const**

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **CharBuffer** & **put** (char value)

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. 472)	*if this buffer's current position is not smaller than its limit*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **CharBuffer** & **put** (int index, char value)

Writes the given char into this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **CharBuffer** * **slice** () **const**

  *Creates a new **CharBuffer** (p. 609) 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. 609) which the caller owns.*

- virtual **lang::CharSequence** * **subSequence** (int start, int end) **const**

  *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. 455) + start, and its limit will be **position()** (p. 455) + end. The new **Buffer** (p. 451) 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. 456).*
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. 456).*

  Returns

  The new character buffer, caller owns.

  Exceptions

IndexOutOfBoundsException	*if the preconditions on start and end fail.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **CharArrayBuffer** (p. 600) as Read-Only.*

Protected Attributes

- **decaf::lang::Pointer** < **ByteArrayAdapter** > **_array**
- int **offset**
- int **length**
- bool **readOnly**

6.111.1 Constructor & Destructor Documentation

6.111.1.1 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (int *size*, bool *readOnly* = `false`)

Creates a **CharArrayBuffer** (p. 600) 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

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArguementException	if the capacity value is negative.

6.111.1.2 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (char * *array*, int *size*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a **CharArrayBuffer** (p. 600) 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	The actual array to wrap.
size	The size of the given array.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if offset is greater than array capacity.

6.111.1.3 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **CharArrayBuffer** (p. 600) will be that of the remaining capacity of the passed buffer.

Parameters

array	The ByteArrayAdapter to wrap.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if array is NULL
IndexOutOfBounds-Exception	if offset + length is greater than array size.

6.111.1.4 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const CharArrayBuffer & *other*)

Create a **CharArrayBuffer** (p. 600) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.

Parameters

other	The **CharArrayBuffer** (p. 600) this one is to mirror.

6.111.1.5 virtual decaf::internal::nio::CharArrayBuffer::~CharArrayBuffer ( ) [virtual]

6.111.2 Member Function Documentation

6.111.2.1 virtual char* decaf::internal::nio::CharArrayBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 613).

6.111.2.2 virtual int decaf::internal::nio::CharArrayBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 614).

6.111.2.3 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. 614).

6.111.2.4 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 609).

Exceptions

ReadOnlyBufferException *(p. 1739)*	- If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 615).

6.111.2.5 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** (p. 451) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 615).

6.111.2.6 virtual char decaf::internal::nio::CharArrayBuffer::get ( ) [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. 474)*	if there no more data to return

Implements **decaf::nio::CharBuffer** (p. 615).

6.111.2.7 virtual char decaf::internal::nio::CharArrayBuffer::get (int *index*) const [virtual]

Absolute get method.

Reads the char at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the char is to be read.

Returns

the char that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit or is negative.

Implements **decaf::nio::CharBuffer** (p. 616).

6.111.2.8 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. Subclasses 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. 617).

6.111.2.9 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.

Implements **decaf::nio::Buffer** (p. 454).

6.111.2.10 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (char *value*) [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. 472)*	if this buffer's current position is not smaller than its limit
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 619).

6.111.2.11 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (int *index*, char *value*) [virtual]

Writes the given char into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The char to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 619).

6.111.2.12 virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **CharArrayBuffer** (p. 600) as Read-Only.

Parameters

value	Boolean value, true if this buffer is to be read-only, false otherwise.

6.111.2.13 virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice ( ) const [virtual]

Creates a new **CharBuffer** (p. 609) 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. 609) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 621).

6.111.2.14 virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence (int *start,* int *end*) const [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. 455) + start, and its limit will be **position()** (p. 455) + end. The new **Buffer** (p. 451) 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. 456).
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. 456).

Returns

The new character buffer, caller owns.

Exceptions

IndexOutOfBounds-Exception	if the preconditions on start and end fail.

Implements **decaf::nio::CharBuffer** (p. 621).

6.111.3 Field Documentation

6.111.3.1 decaf::lang::Pointer<ByteArrayAdapter> decaf::internal::nio::CharArrayBuffer::_array [protected]

6.111.3.2 int decaf::internal::nio::CharArrayBuffer::length [protected]

6.111.3.3 int decaf::internal::nio::CharArrayBuffer::offset [protected]

6.111.3.4 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**

6.112 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)

  Appends the specified character to this buffer.
- **CharBuffer** & **append** (**const lang::CharSequence** *value)

  Appends the specified character sequence to this buffer.
- **CharBuffer** & **append** (**const lang::CharSequence** *value, int start, int end)

  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

  Returns the character array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0

  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** (int index) **const**

  Reads the character at the given index relative to the current position.
- virtual **CharBuffer** & **compact** ()=0

  Compacts this buffer.
- virtual **CharBuffer** * **duplicate** ()=0

  Creates a new char buffer that shares this buffer's content.
- virtual char **get** ()=0

  Relative get method.
- virtual char **get** (int index) **const** =0

  Absolute get method.
- **CharBuffer** & **get** (std::vector< char > buffer)

  Relative bulk get method.
- **CharBuffer** & **get** (char *buffer, int size, int offset, int **length**)

  Relative bulk get method.
- virtual bool **hasArray** () **const** =0

  Tells whether or not this buffer is backed by an accessible char array.
- int **length** () **const**

  Returns the length of this character buffer.
- **CharBuffer** & **put** (**CharBuffer** &src)

  This method transfers the chars remaining in the given source buffer into this buffer.
- **CharBuffer** & **put** (**const** char *buffer, int size, int offset, int **length**)

  This method transfers chars into this buffer from the given source array.
- **CharBuffer** & **put** (std::vector< char > &buffer)

  This method transfers the entire content of the given source char array into this buffer.
- virtual **CharBuffer** & **put** (char value)=0

  Writes the given char into this buffer at the current position, and then increments the position.
- virtual **CharBuffer** & **put** (int index, char value)=0

  Writes the given char into this buffer at the given index.
- **CharBuffer** & **put** (std::string &src, int start, int end)

Relative bulk put method (optional operation).

- **CharBuffer** & **put** (**const** std::string &src)

  Relative bulk put method (optional operation).

- virtual int **read** (**CharBuffer** *target)

  Attempts to read characters into the specified character buffer.

- virtual **lang::CharSequence** * **subSequence** (int start, int end) **const** =0

  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. 609) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (**const CharBuffer** &value) **const**

- virtual bool **equals** (**const CharBuffer** &value) **const**

- virtual bool **operator==** (**const CharBuffer** &value) **const**

- virtual bool **operator**< (**const CharBuffer** &value) **const**

Static Public Member Functions

- static **CharBuffer** * **allocate** (int **capacity**)

  Allocates a new character buffer.

- static **CharBuffer** * **wrap** (char *__array__, int size, int offset, int **length**)

  *Wraps the passed buffer with a new **CharBuffer** (p. 609).*

- static **CharBuffer** * **wrap** (std::vector< char > &buffer)

  *Wraps the passed STL char Vector in a **CharBuffer** (p. 609).*

Protected Member Functions

- **CharBuffer** (int **capacity**)

  *Creates a **CharBuffer** (p. 609) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.112.1 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);
```

6.112.2 Constructor & Destructor Documentation

6.112.2.1 decaf::nio::CharBuffer::CharBuffer (int *capacity*) [protected]

Creates a **CharBuffer** (p. 609) 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	The size of the array, this is the limit we read and write to.

Exceptions

IllegalArguementException	if capacity is negative.

6.112.2.2 virtual decaf::nio::CharBuffer::~CharBuffer ( ) [inline, virtual]

6.112.3 Member Function Documentation

6.112.3.1 static CharBuffer* decaf::nio::CharBuffer::allocate (int *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. 609) that was allocated, caller owns.

Exceptions

IndexOutOfBounds-Exception	if capacity is negative.

6.112.3.2 CharBuffer& decaf::nio::CharBuffer::append (char *value*) [virtual]

Appends the specified character to this buffer.

Parameters

value	The char to append.

Returns

a reference to this modified **CharBuffer** (p. 609).

Exceptions

BufferOverflowException *(p. 472)*	if there is no more space
ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.

Implements **decaf::lang::Appendable** (p. 343).

6.112.3.3 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value*) [virtual]

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. 609)

Exceptions

BufferOverflowException *(p. 472)*	if there is no more space
ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.

Implements **decaf::lang::Appendable** (p. 343).

6.112.3.4 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value,* int *start,* int *end*) [virtual]

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. 609).

Exceptions

BufferOverflowException *(p. 472)*	if there is no more space

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
IndexOutOfBounds-Exception	if start $>$ end, or $>$ length of sequence.

Implements **decaf::lang::Appendable** (p. 344).

6.112.3.5 virtual char* decaf::nio::CharBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 604).

6.112.3.6 virtual int decaf::nio::CharBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 605).

6.112.3.7 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. 605).

6.112.3.8 char decaf::nio::CharBuffer::charAt (int *index*) const [virtual]

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. 455) + index.

Exceptions

IndexOutOfBounds-Exception	if the index + the current position exceeds the size of the buffer or the index is negative.

Implements **decaf::lang::CharSequence** (p. 623).

6.112.3.9 virtual CharBuffer& decaf::nio::CharBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 609).

Exceptions

ReadOnlyBufferException *(p. 1739)*	- If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 605).

6.112.3.10 virtual int decaf::nio::CharBuffer::compareTo (const CharBuffer & *value*) const [virtual]

6.112.3.11 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. 451) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 606).

6.112.3.12 virtual bool decaf::nio::CharBuffer::equals (const CharBuffer & *value*) const [virtual]

6.112.3.13 virtual char decaf::nio::CharBuffer::get ( ) [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. 474)*	if there no more data to return

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 606).

6.112.3.14 virtual char decaf::nio::CharBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the char at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the char is to be read.

Returns

the char that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit or is negative.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 606).

6.112.3.15 CharBuffer& decaf::nio::CharBuffer::get (std::vector< char > *buffer*)

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. 609).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length chars remaining in this buffer.

6.112.3.16 CharBuffer& decaf::nio::CharBuffer::get (char * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no bytes are transferred and a **BufferUnderflowException** (p. 474) 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	The pointer to an allocated buffer to fill.
size	The size of the buffer passed.
offset	The position in the buffer to start filling.
length	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length chars remaining in this buffer
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.112.3.17 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. Subclasses 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. 607).

6.112.3.18 int decaf::nio::CharBuffer::length ( ) const [inline, virtual]

Returns the length of this character buffer.

Returns

the length of this buffer from the position to the limit.

Implements **decaf::lang::CharSequence** (p. 624).

6.112.3.19 virtual bool decaf::nio::CharBuffer::operator< (const CharBuffer & *value*) const [virtual]

6.112.3.20 virtual bool decaf::nio::CharBuffer::operator== (const CharBuffer & *value*) const [virtual]

6.112.3.21 CharBuffer& decaf::nio::CharBuffer::put (CharBuffer & *src*)

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. 456), then no chars are transferred and a **BufferOverflowException** (p. 472) 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. 472)*	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. 1739)*	if this buffer is read-only.

6.112.3.22 CharBuffer& decaf::nio::CharBuffer::put (const char * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no chars are transferred and a **BufferOverflowException** (p. 472) 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.
size	The size of the buffer passed.
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. 472)*	if there is insufficient space in this buffer
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.112.3.23 CharBuffer& decaf::nio::CharBuffer::put (std::vector< char > & *buffer*)

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. 609).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

6.112.3.24 virtual CharBuffer& decaf::nio::CharBuffer::put (char *value*) [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. 472)*	if this buffer's current position is not smaller than its limit
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 607).

6.112.3.25 virtual CharBuffer& decaf::nio::CharBuffer::put (int *index,* char *value*) [pure virtual]

Writes the given char into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The char to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 607).

6.112.3.26 CharBuffer& decaf::nio::CharBuffer::put (std::string & *src,* int *start,* int *end*)

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. 456), then no characters are transferred and a **BufferOverflowException** (p. 472) 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	The position in src to start from.
end	The position in src to stop at.

Returns

a reference to this **CharBuffer** (p. 609).

Exceptions

BufferOverflowException *(p. 472)*	if this buffer's current position is not
IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

6.112.3.27 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*)

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. 609).

Exceptions

BufferOverflowException *(p. 472)*	if this buffer's current position is not.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

6.112.3.28 virtual int decaf::nio::CharBuffer::read (CharBuffer * *target*) [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 **CharBuffer** (p. 609).
ReadOnlyBufferException *(p. 1739)*	if this buffer is in read-only mode.

6.112.3.29 virtual CharBuffer* decaf::nio::CharBuffer::slice ( ) const [pure virtual]

Creates a new **CharBuffer** (p. 609) 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. 609) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 608).

6.112.3.30 virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence (int *start,* int *end*) const [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. 455) + start, and its limit will be **position()** (p. 455) + end. The new **Buffer** (p. 451) 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. 456).
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. 456).

Returns

The new character buffer, caller owns.

Exceptions

IndexOutOfBounds-Exception	if the preconditions on start and end fail.

Implements **decaf::lang::CharSequence** (p. 624).

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 608).

6.112.3.31 virtual std::string decaf::nio::CharBuffer::toString ( ) const [virtual]

Returns

a std::string describing this object

Implements **decaf::lang::CharSequence** (p. 624).

6.112.3.32 static CharBuffer* decaf::nio::CharBuffer::wrap (char * *array,* int *size,* int *offset,* int *length*) [static]

Wraps the passed buffer with a new **CharBuffer** (p. 609).

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.
size	The size of the array passed in.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new **CharBuffer** (p. 609) that is backed by buffer, caller owns.

Exceptions

NullPointerException	if the array pointer is Null.
IndexOutOfBounds-Exception	if capacity is negative.

6.112.3.33 static CharBuffer* decaf::nio::CharBuffer::wrap (std::vector< char > & *buffer*) [static]

Wraps the passed STL char Vector in a **CharBuffer** (p. 609).

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. 609) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**CharBuffer.h**

6.113 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 623) 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 int **length** () **const** =0
- virtual char **charAt** (int index) **const** =0

  Returns the Char at the specified index so long as the index is not greater than the length of the sequence.
- virtual **CharSequence** * **subSequence** (int start, int end) **const** =0

  *Returns a new **CharSequence** (p. 623) that is a subsequence of this sequence.*
- virtual std::string **toString** () **const** =0

6.113.1 Detailed Description

A **CharSequence** (p. 623) 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. 623) 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.

6.113.2 Constructor & Destructor Documentation

6.113.2.1 virtual decaf::lang::CharSequence::~CharSequence ( ) [inline, virtual]

6.113.3 Member Function Documentation

6.113.3.1 virtual char decaf::lang::CharSequence::charAt (int *index*) const [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	The position to return the char at.

Returns

the char at the given position.

Exceptions

IndexOutOfBounds-Exception	if index is $>$ than **length()** (p. 624) or negative

Implemented in **decaf::nio::CharBuffer** (p. 614), and **decaf::lang::String** (p. 2033).

6.113.3.2 virtual int decaf::lang::CharSequence::length ( ) const [pure virtual]

Returns

the length of the underlying character sequence.

Implemented in **decaf::nio::CharBuffer** (p. 617), and **decaf::lang::String** (p. 2034).

6.113.3.3 virtual CharSequence* decaf::lang::CharSequence::subSequence (int *start,* int *end*) const [pure virtual]

Returns a new **CharSequence** (p. 623) 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. 623)

Exceptions

IndexOutOfBounds-Exception	if start or end > **length()** (p. 624) or start or end are negative.

Implemented in **decaf::nio::CharBuffer** (p. 621), **decaf::internal::nio::CharArrayBuffer** (p. 608), and **decaf::lang::String** (p. 2034).

6.113.3.4 virtual std::string decaf::lang::CharSequence::toString ( ) const [pure virtual]

Returns

the string representation of this **CharSequence** (p. 623)

Implemented in **decaf::lang::String** (p. 2034), and **decaf::nio::CharBuffer** (p. 622).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**CharSequence.h**

6.114 decaf::util::zip::CheckedInputStream Class Reference

An implementation of a FilterInputStream that will maintain a **Checksum** (p. 628) of the bytes read, the **Checksum** (p. 628) can then be used to verify the integrity of the input stream.

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

Inheritance diagram for decaf::util::zip::CheckedInputStream:

Public Member Functions

- **CheckedInputStream** (InputStream *__inputStream__, **Checksum** *sum, bool **own**=false)

  *Create a new instance of a **CheckedInputStream** (p. 624).*
- virtual ~**CheckedInputStream** ()
- **Checksum** * **getChecksum** () **const**

  *Returns a Pointer to the **Checksum** (p. 628) that is in use by this **CheckedInputStream** (p. 624).*
- virtual long long **skip** (long long num)

  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.
  *The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)	*if an I/O error occurs.*
UnsupportedOperation-Exception	*if the concrete stream class does not support skipping bytes.*

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.114.1 Detailed Description

An implementation of a FilterInputStream that will maintain a **Checksum** (p. 628) of the bytes read, the **Checksum** (p. 628) can then be used to verify the integrity of the input stream.

Since

1.0

6.114.2 Constructor & Destructor Documentation

6.114.2.1 decaf::util::zip::CheckedInputStream::CheckedInputStream (InputStream * *inputStream,* Checksum * *sum,* bool *own* = `false`)

Create a new instance of a **CheckedInputStream** (p. 624).

Parameters

inputStream	The InputStream instance to Wrap.
sum	The **Checksum** (p. 628) instance to use (does not take ownership of the Pointer).
own	Indicates if this filer should take ownership of the InputStream.

Exceptions

NullPointerException	if the **Checksum** (p. 628) pointer is NULL.

6.114.2.2 virtual decaf::util::zip::CheckedInputStream::~CheckedInputStream ( ) [virtual]

6.114.3 Member Function Documentation

6.114.3.1 virtual int decaf::util::zip::CheckedInputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.114.3.2 virtual int decaf::util::zip::CheckedInputStream::doReadByte ( ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.114.3.3 Checksum* decaf::util::zip::CheckedInputStream::getChecksum ( ) const [inline]

Returns a Pointer to the **Checksum** (p. 628) that is in use by this **CheckedInputStream** (p. 624).

Returns

the pointer to the **Checksum** (p. 628) instance that is in use by this object.

6.114.3.4 virtual long long decaf::util::zip::CheckedInputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Adds the skipped bytes into the **Checksum** (p. 628).

Reimplemented from **decaf::io::FilterInputStream** (p. 1036).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CheckedInputStream.h**

6.115 decaf::util::zip::CheckedOutputStream Class Reference

An implementation of a FilterOutputStream that will maintain a **Checksum** (p. 628) of the bytes written, the **Checksum** (p. 628) can then be used to verify the integrity of the output stream.

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

Inheritance diagram for decaf::util::zip::CheckedOutputStream:

Public Member Functions

- **CheckedOutputStream** (**decaf::io::OutputStream** ∗**outputStream**, **Checksum** ∗sum, bool **own**=false)

  *Create a new instance of a **CheckedOutputStream** (p. 627).*
- virtual ∼**CheckedOutputStream** ()
- **Checksum** ∗ **getChecksum** () **const**

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (**const** unsigned char ∗buffer, int size, int offset, int length)

6.115.1 Detailed Description

An implementation of a FilterOutputStream that will maintain a **Checksum** (p. 628) of the bytes written, the **Checksum** (p. 628) can then be used to verify the integrity of the output stream.

Since

1.0

6.115.2 Constructor & Destructor Documentation

6.115.2.1 decaf::util::zip::CheckedOutputStream::CheckedOutputStream (decaf::io::OutputStream * *outputStream,* Checksum * *sum,* bool *own* = false)

Create a new instance of a **CheckedOutputStream** (p. 627).

Parameters

outputStream	The OutputStream instance to Wrap.
sum	The **Checksum** (p. 628) instance to use (does not take ownership of the Pointer).
own	Indicates if this filer should take ownership of the InputStream.

Exceptions

NullPointerException	if the **Checksum** (p. 628) pointer is NULL.

6.115.2.2 virtual decaf::util::zip::CheckedOutputStream::~CheckedOutputStream ( ) [virtual]

6.115.3 Member Function Documentation

6.115.3.1 virtual void decaf::util::zip::CheckedOutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.115.3.2 virtual void decaf::util::zip::CheckedOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.115.3.3 Checksum* decaf::util::zip::CheckedOutputStream::getChecksum ( ) const [inline]

Returns

a pointer to the **Checksum** (p. 628) instance in use by this object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CheckedOutputStream.h**

6.116 decaf::util::zip::Checksum Class Reference

An interface used to represent **Checksum** (p. 628) values in the Zip package.

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

Inheritance diagram for decaf::util::zip::Checksum:

Public Member Functions

- virtual ~**Checksum** ()
- virtual long long **getValue** () **const** =0
- virtual void **reset** ()=0

  Reset the checksum to its initial value.
- virtual void **update** (**const** std::vector< unsigned char > &buffer)=0

  Updates the current checksum with the specified vector of bytes.
- virtual void **update** (**const** std::vector< unsigned char > &buffer, int offset, int length)=0

  Updates the current checksum with the specified array of bytes.
- virtual void **update** (**const** unsigned char *buffer, int size, int offset, int length)=0

  Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)=0

  Updates the current checksum with the specified byte value.

6.116.1 Detailed Description

An interface used to represent **Checksum** (p. 628) values in the Zip package.

Since

1.0

6.116.2 Constructor & Destructor Documentation

6.116.2.1 virtual decaf::util::zip::Checksum::~Checksum ( ) [inline, virtual]

6.116.3 Member Function Documentation

6.116.3.1 virtual long long decaf::util::zip::Checksum::getValue ( ) const [pure virtual]

Returns

the current checksum value.

Implemented in **decaf::util::zip::Adler32** (p. 341), and **decaf::util::zip::CRC32** (p. 826).

6.116.3.2 virtual void decaf::util::zip::Checksum::reset ( ) [pure virtual]

Reset the checksum to its initial value.

Implemented in **decaf::util::zip::Adler32** (p. 341), and **decaf::util::zip::CRC32** (p. 826).

6.116.3.3 virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & *buffer*) [pure virtual]

Updates the current checksum with the specified vector of bytes.

Parameters

buffer	The buffer to read the updated bytes from.

Implemented in **decaf::util::zip::Adler32** (p. 341), and **decaf::util::zip::CRC32** (p. 827).

6.116.3.4 virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & *buffer,* int *offset,* int *length*) [pure virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer	The buffer to read the updated bytes from.
offset	The position in the buffer to start reading.
length	The amount of data to read from the byte buffer.

Exceptions

IndexOutOfBounds-Exception	if offset + length > size of the buffer.

Implemented in **decaf::util::zip::Adler32** (p. 341), and **decaf::util::zip::CRC32** (p. 827).

6.116.3.5 virtual void decaf::util::zip::Checksum::update (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [pure virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer	The buffer to read the updated bytes from.
size	The size of the passed buffer.
offset	The position in the buffer to start reading.
length	The amount of data to read from the byte buffer.

Exceptions

NullPointerException	if the passed buffer is NULL.
IndexOutOfBounds-Exception	if offset + length > size of the buffer.

Implemented in **decaf::util::zip::Adler32** (p. 342), and **decaf::util::zip::CRC32** (p. 827).

6.116.3.6 virtual void decaf::util::zip::Checksum::update (int *byte*) [pure virtual]

Updates the current checksum with the specified byte value.

Parameters

byte	The byte value to update the current **Checksum** (p. 628) with (0..255).

Implemented in **decaf::util::zip::Adler32** (p. 342), and **decaf::util::zip::CRC32** (p. 827).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Checksum.h**

6.117 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. 990).*
- **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 ()

6.117.1 Constructor & Destructor Documentation

6.117.1.1 decaf::lang::exceptions::ClassCastException::ClassCastException ( ) throw () [inline]

Default Constructor.

6.117.1.2 decaf::lang::exceptions::ClassCastException::ClassCastException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.117.1.3 decaf::lang::exceptions::ClassCastException::ClassCastException (const ClassCastException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.117.1.4 decaf::lang::exceptions::ClassCastException::ClassCastException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.117.1.5 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

6.117.1.6 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

6.117.1.7 virtual decaf::lang::exceptions::ClassCastException::~ClassCastException ( ) throw () [inline, virtual]

6.117.2 Member Function Documentation

6.117.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**ClassCastException.h**

6.118 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** () throw ()
- virtual void **close** ()=0

  Closes this object and deallocates the appropriate resources.

6.118.1 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

6.118.2 Constructor & Destructor Documentation

6.118.2.1 virtual cms::Closeable::~Closeable ( ) throw () [virtual]

6.118.3 Member Function Documentation

6.118.3.1 virtual void cms::Closeable::close ( ) [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

CMSException *(p. 640)*	- If an error occurs while the resource is being closed.

Implemented in **activemq::core::ActiveMQConnection** (p. 152), **activemq::core::ActiveMQSession** (p. 262), **cms::Session** (p. 1833), **activemq::core::ActiveMQProducer** (p. 240), **activemq::core::ActiveMQConsumer** (p. 184), **activemq::cmsutil::PooledSession** (p. 1623), **activemq::core::ActiveMQQueueBrowser** (p. 253), **cms::Connection** (p. 726), **activemq::commands::ActiveMQTempDestination** (p. 295), **activemq::cmsutil::CachedConsumer** (p. 570), and **activemq::cmsutil::CachedProducer** (p. 574).

The documentation for this class was generated from the following file:

- src/main/cms/**Closeable.h**

6.119 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** ()=0

  Closes this object and deallocates the appropriate resources.

6.119.1 Detailed Description

Interface for a class that implements the close method.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 virtual decaf::io::Closeable::~Closeable ( ) [inline, virtual]

6.119.3 Member Function Documentation

6.119.3.1 virtual void decaf::io::Closeable::close ( ) [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(*p. *1198)*	if an error occurs while closing.

Implemented in **decaf::net::Socket** (p. 1905), **activemq::transport::IOTransport** (p. 1202), **activemq::transport::mock::MockTransport** (p. 1511), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1597), **activemq::transport::TransportFilter** (p. 2170), **activemq::transport::failover::FailoverTransport** (p. 1013), **decaf::util::zip::InflaterInputStream** (p. 1131), **decaf::util::zip::DeflaterOutputStream** (p. 923), **activemq::transport::tcp::TcpTransport** (p. 2087), **decaf::util::logging::StreamHandler** (p. 2019), **decaf::io::FilterOutputStream** (p. 1039), **activemq::transport::correlator::ResponseCorrelator** (p. 1786), **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1565), **decaf::io::BufferedInputStream** (p. 459), **decaf::io::BlockingByteArrayInputStream** (p. 416), **decaf::io::FilterInputStream** (p. 1034), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2084), **activemq::transport::inactivity::InactivityMonitor** (p. 1104), **decaf::io::InputStreamReader** (p. 1144), **decaf::io::OutputStreamWriter** (p. 1607), **decaf::util::logging::ConsoleHandler** (p. 769), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 1581), **decaf::io::InputStream** (p. 1136), **decaf::io::OutputStream** (p. 1602), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 2086), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 1583), **decaf::internal::io::StandardErrorOutputStream** (p. 1960), and **decaf::internal::io::StandardOutputStream** (p. 1962).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Closeable.h**

6.120 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. 2161) 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.

6.120.1 Constructor & Destructor Documentation

6.120.1.1 activemq::transport::failover::CloseTransportsTask::CloseTransportsTask ( )

6.120.1.2 virtual activemq::transport::failover::CloseTransportsTask::~CloseTransportsTask ( ) [virtual]

6.120.2 Member Function Documentation

6.120.2.1 void activemq::transport::failover::CloseTransportsTask::add (const Pointer< Transport > & *transport*)

Add a new **Transport** (p. 2161) to close.

6.120.2.2 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. 692).

6.120.2.3 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. 2073).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**CloseTransportsTask.h**

6.121 activemq::cmsutil::CmsAccessor Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p. 649) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 741) 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

- **CmsAccessor** (**const CmsAccessor** &)
- **CmsAccessor** & **operator=** (**const CmsAccessor** &)
- virtual void **init** ()

  Initializes this object and prepares it for use.
- virtual void **destroy** ()

  Shuts down this object and destroys any allocated resources.
- virtual **cms::Connection** * **createConnection** ()

  Create a CMS Connection via this template's ConnectionFactory.
- virtual **cms::Session** * **createSession** (**cms::Connection** *con)

  Create a CMS Session for the given Connection.
- virtual void **checkConnectionFactory** ()

  Verifies that the connection factory is valid.

6.121.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p. 649) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 741) to operate on.

The subclass **activemq.cmsutil.CmsDestinationAccessor** (p. 638) adds further, destination-related properties.

Not intended to be used directly.

See also

activemq.cmsutil.CmsDestinationAccessor (p. 638)
activemq.cmsutil.CmsTemplate (p. 649)

6.121.2 Constructor & Destructor Documentation

6.121.2.1 activemq::cmsutil::CmsAccessor::CmsAccessor (const CmsAccessor &) [protected]

6.121.2.2 activemq::cmsutil::CmsAccessor::CmsAccessor ( )

6.121.2.3 virtual activemq::cmsutil::CmsAccessor::~CmsAccessor ( ) [virtual]

6.121.3 Member Function Documentation

6.121.3.1 virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory ( ) [protected, virtual]

Verifies that the connection factory is valid.

Exceptions

IllegalStateException	if this object has not been initialized.

6.121.3.2 virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection ( ) [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

Returns

the new CMS Connection

Exceptions

CMSException	if thrown by CMS API methods
IllegalStateException	if this object has not been initialized.

6.121.3.3 virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession (cms::Connection * *con*) [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

CMSException	if thrown by CMS API methods
IllegalStateException	if this object has not been initialized.

6.121.3.4 virtual void activemq::cmsutil::CmsAccessor::destroy ( ) [inline, protected, virtual]

Shuts down this object and destroys any allocated resources.

Exceptions

CMSException	if an error occurs during destruction.
IllegalStateException	if this object has already been destroyed.

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 651), and **activemq::cmsutil::CmsDestinationAccessor** (p. 639).

6.121.3.5 virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory ( ) const [inline, virtual]

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.121.3.6 virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory ( ) [inline, virtual]

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.121.3.7 virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager ( ) [inline, virtual]

6.121.3.8 virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager ( ) const [inline, virtual]

6.121.3.9 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

6.121.3.10 virtual void activemq::cmsutil::CmsAccessor::init ( ) [protected, virtual]

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Exceptions

CMSException	if an error occurs during initialization.
IllegalStateException	if this object has already been initialized.

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 654), and **activemq::cmsutil::CmsDestinationAccessor** (p. 640).

6.121.3.11 CmsAccessor& activemq::cmsutil::CmsAccessor::operator= (const CmsAccessor &) [protected]

6.121.3.12 virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory (cms::ConnectionFactory * *connectionFactory*) [inline, virtual]

Set the ConnectionFactory to use for obtaining CMS Connections.

6.121.3.13 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

session-Acknowledge-Mode	The acknowledgment mode to assign to the Session.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsAccessor.h**

6.122 activemq::cmsutil::CmsDestinationAccessor Class Reference

Extends the **CmsAccessor** (p. 635) 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** ()

  Initializes this object and prepares it for use.
- virtual void **destroy** ()

  Shuts down this object and destroys any allocated resources.
- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, **const** std::string &destName)
- virtual void **checkDestinationResolver** ()

6.122.1 Detailed Description

Extends the **CmsAccessor** (p. 635) to add support for resolving destination names.

Not intended to be used directly.

See also

CmsTemplate (p. 649)
CmsAccessor (p. 635)

6.122.2 Constructor & Destructor Documentation

6.122.2.1 activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ( )

6.122.2.2 virtual activemq::cmsutil::CmsDestinationAccessor::~CmsDestinationAccessor ( ) [virtual]

6.122.3 Member Function Documentation

6.122.3.1 virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver ( ) [protected, virtual]

6.122.3.2 virtual void activemq::cmsutil::CmsDestinationAccessor::destroy ( ) [protected, virtual]

Shuts down this object and destroys any allocated resources.

Exceptions

CMSException	if an error occurs during destruction.
IllegalStateException	if this object has already been destroyed.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 637).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 651).

6.122.3.3 virtual DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ( ) [inline, virtual]

6.122.3.4 virtual const DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ( ) const [inline, virtual]

6.122.3.5 virtual void activemq::cmsutil::CmsDestinationAccessor::init ( ) [protected, virtual]

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Exceptions

CMSException	if an error occurs during initialization.
IllegalStateException	if this object has already been initialized.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 638).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 654).

6.122.3.6 virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain ( ) const [inline, virtual]

6.122.3.7 virtual cms::Destination* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName (cms::Session * *session,* const std::string & *destName*) [protected, virtual]

6.122.3.8 virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (DestinationResolver * *destRes*) [inline, virtual]

6.122.3.9 virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 659).

Referenced by activemq::cmsutil::CmsTemplate::setPubSubDomain().

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsDestinationAccessor.h**

6.123 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

- **CMSException** ()
- **CMSException** (**const CMSException** &ex)
- **CMSException** (**const** std::string &message)
- **CMSException** (**const** std::string &message, **const** std::exception *cause)
- **CMSException** (**const** std::string &message, **const** std::exception *cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**CMSException** () 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. 643) function to return the cause of the exception.*

6.123.1 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 **CMSException** (p. 640). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSException** (p. 640) instances.

Since

1.0

6.123.2 Constructor & Destructor Documentation

6.123.2.1 cms::CMSException::CMSException ( )

6.123.2.2 cms::CMSException::CMSException (const CMSException & *ex*)

6.123.2.3 cms::CMSException::CMSException (const std::string & *message*)

6.123.2.4 cms::CMSException::CMSException (const std::string & *message,* const std::exception * *cause*)

6.123.2.5 cms::CMSException::CMSException (const std::string & *message,* const std::exception * *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.123.2.6 virtual cms::CMSException::~CMSException ( ) throw () [virtual]

6.123.3 Member Function Documentation

6.123.3.1 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.

6.123.3.2 virtual std::string cms::CMSException::getMessage ( ) const [virtual]

Gets the cause of the error.

Returns

string errors message

6.123.3.3 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

6.123.3.4 virtual std::string cms::CMSException::getStackTraceString ( ) const [virtual]

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

6.123.3.5 virtual void cms::CMSException::printStackTrace ( ) const [virtual]

Prints the stack trace to std::err.

6.123.3.6 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.

6.123.3.7 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**).

6.123.3.8 virtual const char* cms::CMSException::what ( ) const throw () [virtual]

Overloads the std::exception **what()** (p. 643) 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**

6.124 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)

6.124.1 Constructor & Destructor Documentation

6.124.1.1 virtual activemq::util::CMSExceptionSupport::~CMSExceptionSupport ( ) [virtual]

6.124.2 Member Function Documentation

6.124.2.1 static cms::CMSException activemq::util::CMSExceptionSupport::create (const std::string & *msg,* const decaf::lang::Exception & *cause*) [static]

6.124.2.2 static cms::CMSException activemq::util::CMSExceptionSupport::create (const decaf::lang::Exception & *cause*) [static]

6.124.2.3 static cms::MessageEOFException activemq::util::CMSExceptionSupport-::createMessageEOFException (const decaf::lang::Exception & *cause*) [static]

6.124.2.4 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**

6.125 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** () throw ()
- virtual int **size** () **const** =0

  Returns the current count of all the Properties that are currently stored in the Properties object.
- 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 std::string **remove** (**const** std::string &name)=0

  Removes the property with the given name.
- virtual std::vector< std::string > **propertyNames** () **const** =0

  Returns a vector containing all the names of the properties currently stored in the Properties object.
- 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.

6.125.1 Detailed Description

Interface for a Java-like properties object.

This is essentially a map of key-value string pairs.

Since

1.1

6.125.2 Constructor & Destructor Documentation

6.125.2.1 virtual cms::CMSProperties::~CMSProperties ( ) throw () [virtual]

6.125.3 Member Function Documentation

6.125.3.1 virtual void cms::CMSProperties::clear ( ) [pure virtual]

Clears all properties from the map.

Implemented in **activemq::util::ActiveMQProperties** (p. 246).

6.125.3.2 virtual CMSProperties* cms::CMSProperties::clone ( ) const [pure virtual]

Clones this object.

Returns

a replica of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 246).

6.125.3.3 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.

6.125.3.4 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. 246).

6.125.3.5 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. 246).

6.125.3.6 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. 247).

6.125.3.7 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. 247).

6.125.3.8 virtual std::vector<std::string> cms::CMSProperties::propertyNames ( ) const [pure virtual]

Returns a vector containing all the names of the properties currently stored in the Properties object.

Returns

an STL std::vector<std::string> with all the currently stored property names.

Implemented in **activemq::util::ActiveMQProperties** (p. 247).

6.125.3.9 virtual std::string cms::CMSProperties::remove (const std::string & *name*) [pure virtual]

Removes the property with the given name.

If the property existed in the collection then it is removed and returned, otherwise an empty string is returned.

Parameters

name	the name of the property to be removed.

Returns

the value that was removed from the Properties, or empty string.

Implemented in **activemq::util::ActiveMQProperties** (p. 247).

6.125.3.10 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. 248).

6.125.3.11 virtual int cms::CMSProperties::size ( ) const [pure virtual]

Returns the current count of all the Properties that are currently stored in the Properties object.

Returns

the number of properties currently stored.

Implemented in **activemq::util::ActiveMQProperties** (p. 248).

6.125.3.12 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. 248).

6.125.3.13 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. 248).

The documentation for this class was generated from the following file:

- src/main/cms/**CMSProperties.h**

6.126 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** ()
- **CMSSecurityException** (**const CMSSecurityException** &ex)
- **CMSSecurityException** (**const** std::string &message)
- **CMSSecurityException** (**const** std::string &message, **const** std::exception ∗cause)
- **CMSSecurityException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ∼**CMSSecurityException** () throw ()

6.126.1 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

6.126.2 Constructor & Destructor Documentation

6.126.2.1 cms::CMSSecurityException::CMSSecurityException ( )

6.126.2.2 cms::CMSSecurityException::CMSSecurityException (const CMSSecurityException & *ex*)

6.126.2.3 cms::CMSSecurityException::CMSSecurityException (const std::string & *message*)

6.126.2.4 cms::CMSSecurityException::CMSSecurityException (const std::string & *message,* const std::exception ∗ *cause*)

6.126.2.5 cms::CMSSecurityException::CMSSecurityException (const std::string & *message,* const std::exception ∗ *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.126.2.6 virtual cms::CMSSecurityException::∼CMSSecurityException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**CMSSecurityException.h**

6.127 activemq::cmsutil::CmsTemplate Class Reference

CmsTemplate (p. 649) 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** () throw ()
- 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)

  Executes the given action within a CMS Session.
- virtual void **execute** (**ProducerCallback** ∗action)

  Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (**cms::Destination** ∗dest, **ProducerCallback** ∗action)

  Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (**const** std::string &destinationName, **ProducerCallback** ∗action)

  Executes the given action and provides it with a CMS Session and producer.
- virtual void **send** (**MessageCreator** ∗messageCreator)

Convenience method for sending a message to the default destination.

- virtual void **send** (**cms::Destination** ∗dest, **MessageCreator** ∗messageCreator)

  Convenience method for sending a message to the specified destination.

- virtual void **send** (**const** std::string &destinationName, **MessageCreator** ∗messageCreator)

  Convenience method for sending a message to the specified destination.

- virtual **cms::Message** ∗ **receive** ()

  Performs a synchronous read from the default destination.

- virtual **cms::Message** ∗ **receive** (**cms::Destination** ∗destination)

  Performs a synchronous read from the specified destination.

- virtual **cms::Message** ∗ **receive** (**const** std::string &destinationName)

  Performs a synchronous read from the specified destination.

- virtual **cms::Message** ∗ **receiveSelected** (**const** std::string &selector)

  Performs a synchronous read consuming only messages identified by the given selector.

- virtual **cms::Message** ∗ **receiveSelected** (**cms::Destination** ∗destination, **const** std::string &selector)

  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)

  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**

  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**

  Timeout value indicating a blocking receive without timeout.

- static **const** int **DEFAULT_PRIORITY**

  Default message priority.

- static **const** long long **DEFAULT_TIME_TO_LIVE**

  My default, messages should live forever.

Protected Member Functions

- void **init** ()

  Initializes this object and prepares it for use.

- void **destroy** ()

  Shuts down this object and destroys any allocated resources.

Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

6.127.1 Detailed Description

CmsTemplate (p. 649) 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. 649) 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. 649) allows the user to get access to a CMS Session through a user-defined **SessionCallback** (p. 1842). Similarly, if the user wants direct access to a CMS MessageProducer, it can provide a **ProducerCallback** (p. 1689). 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. 1842)
ProducerCallback (p. 1689)
MessageCreator (p. 1458)

6.127.2 Constructor & Destructor Documentation

6.127.2.1 activemq::cmsutil::CmsTemplate::CmsTemplate ( )

6.127.2.2 activemq::cmsutil::CmsTemplate::CmsTemplate (cms::ConnectionFactory * *connectionFactory*)

6.127.2.3 virtual activemq::cmsutil::CmsTemplate::~CmsTemplate ( ) throw () [virtual]

6.127.3 Member Function Documentation

6.127.3.1 void activemq::cmsutil::CmsTemplate::destroy ( ) [protected, virtual]

Shuts down this object and destroys any allocated resources.

Exceptions

CMSException	if an error occurs during destruction.
IllegalStateException	if this object has already been destroyed.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 639).

6.127.3.2 virtual void activemq::cmsutil::CmsTemplate::execute (SessionCallback * *action*) [virtual]

Executes the given action within a CMS Session.

Parameters

action	the action to perform within a CMS Session

Exceptions

cms::CMSException *(p. 640)*	thrown if an error occurs.

6.127.3.3 virtual void activemq::cmsutil::CmsTemplate::execute (ProducerCallback * action) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

action	the action to perform

Exceptions

cms::CMSException *(p. 640)*	thrown if an error occurs.

6.127.3.4 virtual void activemq::cmsutil::CmsTemplate::execute (cms::Destination * *dest,* ProducerCallback * *action*) [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. 640)*	thrown if an error occurs.

6.127.3.5 virtual void activemq::cmsutil::CmsTemplate::execute (const std::string & *destinationName,* ProducerCallback * *action*) [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. 640)*	thrown if an error occurs.

6.127.3.6 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.

6.127.3.7 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.

6.127.3.8 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.

6.127.3.9 virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode ( ) const [inline, virtual]

Return the delivery mode to use when sending a message.

6.127.3.10 virtual int activemq::cmsutil::CmsTemplate::getPriority ( ) const [inline, virtual]

Return the priority of a message when sending.

6.127.3.11 virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout ( ) const [inline, virtual]

6.127.3.12 virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive ( ) const [inline, virtual]

Return the time-to-live of the message when sending.

6.127.3.13 void activemq::cmsutil::CmsTemplate::init ( ) [protected, virtual]

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Exceptions

CMSException	if an error occurs during initialization.
IllegalStateException	if this object has already been initialized.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 640).

6.127.3.14 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. 658)
setPriority (p. 658)
setTimeToLive (p. 659)

6.127.3.15 virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled ( ) const [inline, virtual]

6.127.3.16 virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled ( ) const [inline, virtual]

6.127.3.17 virtual bool activemq::cmsutil::CmsTemplate::isNoLocal ( ) const [inline, virtual]

6.127.3.18 virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ( ) [virtual]

Performs a synchronous read from the default destination.

Returns

the message

Exceptions

cms::CMSException *(p. 640)*	thrown if an error occurs

6.127.3.19 virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (cms::Destination * *destination*) [virtual]

Performs a synchronous read from the specified destination.

Parameters

destination	the destination to receive on

Returns

the message

Exceptions

cms::CMSException *(p. 640)*	thrown if an error occurs

6.127.3.20 virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (const std::string & *destinationName*) [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::CMSException *(p. 640)*	thrown if an error occurs

6.127.3.21 virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & *selector*) [virtual]

Performs a synchronous read consuming only messages identified by the given selector.

Parameters

selector	the selector expression.

Returns

the message

Exceptions

cms::CMSException *(p. 640)*	thrown if an error occurs

6.127.3.22 virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (cms::Destination * *destination,* const std::string & *selector*) [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. 640)*	thrown if an error occurs

6.127.3.23 virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & *destinationName,* const std::string & *selector*) [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. 640)*	thrown if an error occurs

6.127.3.24 virtual void activemq::cmsutil::CmsTemplate::send (MessageCreator * *messageCreator*) [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. 640)*	thrown if an error occurs.

6.127.3.25 virtual void activemq::cmsutil::CmsTemplate::send (cms::Destination * *dest,* MessageCreator * *messageCreator*) [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. 640)*	thrown if an error occurs.

6.127.3.26 virtual void activemq::cmsutil::CmsTemplate::send (const std::string & *destinationName,* MessageCreator * *messageCreator*) [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. 640)*	thrown if an error occurs.

6.127.3.27 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

default-Destination	the default destination

6.127.3.28 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

default-Destination-Name	the name of the destination for send/receive to by default.

6.127.3.29 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. 654)

6.127.3.30 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. 658)

6.127.3.31 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. 658)
setPriority (p. 658)
setTimeToLive (p. 659)

6.127.3.32 virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled (bool *messageIdEnabled*) [inline, virtual]

6.127.3.33 virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled (bool *messageTimestampEnabled*) [inline, virtual]

6.127.3.34 virtual void activemq::cmsutil::CmsTemplate::setNoLocal (bool *noLocal*) [inline, virtual]

6.127.3.35 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. 654)

6.127.3.36 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. 640).

References activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain().

6.127.3.37 virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout (long long *receiveTimeout*) [inline, virtual]

6.127.3.38 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. 654)

6.127.4 Friends And Related Function Documentation

6.127.4.1 friend class ProducerExecutor [friend]

6.127.4.2 friend class ReceiveExecutor [friend]

6.127.4.3 friend class ResolveProducerExecutor [friend]

6.127.4.4 friend class ResolveReceiveExecutor [friend]

6.127.4.5 friend class SendExecutor [friend]

6.127.5 Field Documentation

6.127.5.1 const int activemq::cmsutil::CmsTemplate::DEFAULT_PRIORITY [static]

Default message priority.

6.127.5.2 const long long activemq::cmsutil::CmsTemplate::DEFAULT_TIME_TO_LIVE [static]

My default, messages should live forever.

6.127.5.3 const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_INDEFINITE_WAIT [static]

Timeout value indicating a blocking receive without timeout.

6.127.5.4 const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_NO_WAIT [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**

6.128 code Struct Reference

`#include <src/main/decaf/internal/util/zip/inftrees.h>`

Data Fields

- unsigned char **op**
- unsigned char **bits**
- unsigned short **val**

6.128.1 Field Documentation

6.128.1.1 unsigned char code::bits

6.128.1.2 unsigned char code::op

6.128.1.3 unsigned short code::val

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**inftrees.h**

6.129 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 void **copy** (**const Collection**< E > &collection)=0

  *Renders this **Collection*** (p. *660) as a Copy of the given **Collection*** (p. *660).*
- virtual bool **add** (**const** E &value)=0

  Returns true if this collection changed as a result of the call.
- virtual bool **addAll** (**const Collection**< E > &collection)=0

  Adds all of the elements in the specified collection to this collection.
- virtual void **clear** ()=0

  Removes all of the elements from this collection (optional operation).
- virtual bool **contains** (**const** E &value) **const** =0

  Returns true if this collection contains the specified element.
- virtual bool **containsAll** (**const Collection**< E > &collection) **const** =0

  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

  Removes a single instance of the specified element from the collection.
- virtual bool **removeAll** (**const Collection**< E > &collection)=0

  Removes all this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (**const Collection**< E > &collection)=0

  Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual int **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.

6.129.1 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. 660) implementation classes (which typically implement **Collection** (p. 660) 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. 660), 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. 660) 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

6.129.2 Constructor & Destructor Documentation

6.129.2.1 template<typename E> virtual decaf::util::Collection< E >::~Collection ( ) [inline, virtual]

6.129.3 Member Function Documentation

6.129.3.1 template<typename E> virtual bool decaf::util::Collection< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Implemented in **decaf::util::StlList< E >** (p. 1970), **decaf::util::AbstractList< E >** (p. 98), **decaf::util::AbstractList< Pointer< Transport > >** (p. 98), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 98), **decaf::util::AbstractList< CompositeTask * >** (p. 98), **decaf::util::AbstractList< URI >** (p. 98), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 98), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 98), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 98), **decaf::util::AbstractList< Pointer< Command > >** (p. 98), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 98), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 98), **decaf::util::AbstractList< cms::Destination * >** (p. 98), **decaf::util::AbstractList< cms::Session * >** (p. 98), **decaf::util::AbstractList< cms::Connection * >** (p. 98), **decaf::util::PriorityQueue< E >** (p. 1678), **decaf::util::StlSet< E >** (p. 1997), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 1997), **decaf::util::StlSet< Resource * >** (p. 1997), **decaf::util::ArrayList< E >** (p. 348), **decaf::util::LinkedList< E >** (p. 1271), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1271), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1271), **decaf::util::LinkedList< CompositeTask * >** (p. 1271), **decaf::util::LinkedList< URI >** (p. 1271), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1271), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1271), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1271), **decaf::util::LinkedList< Pointer< Command > >** (p. 1271), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1271), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1271), **decaf::util::LinkedList< cms::Destination * >** (p. 1271), **decaf::util::LinkedList< cms::Session * >** (p. 1271), **decaf::util::LinkedList< cms::Connection * >** (p. 1271), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 800), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 800), **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 816), and **decaf::util::AbstractQueue< E >** (p. 109).

Referenced by decaf::util::concurrent::SynchronousQueue< E >::drainTo(), and decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo().

6.129.3.2 template<typename E> virtual bool decaf::util::Collection< E >::addAll (const Collection< E > & *collection*) [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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::StlList< E >** (p. 1971), **decaf::util::ArrayList< E >** (p. 349), **decaf::util::AbstractCollection< E >** (p. 87), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 87), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 87), **decaf::util::AbstractCollection< Resource * >** (p. 87), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 87), **decaf::util::AbstractCollection< CompositeTask * >** (p. 87), **decaf::util::AbstractCollection< URI >** (p. 87), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 87), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 87), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 87), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 87), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 87), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 87), **decaf::util::AbstractCollection< cms::Destination * >** (p. 87), **decaf::util::AbstractCollection< cms::Session * >** (p. 87), **decaf::util::AbstractCollection< cms::Connection * >** (p. 87), **decaf::util::LinkedList< E >** (p. 1272), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1272), **decaf::util::LinkedList< CompositeTask * >** (p. 1272), **decaf::util::LinkedList< URI >** (p. 1272), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1272), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1272), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1272), **decaf::util::LinkedList< Pointer< Command > >** (p. 1272), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1272), **decaf::util::LinkedList< cms::Destination * >** (p. 1272), **decaf::util::LinkedList< cms::Session * >** (p. 1272), **decaf::util::LinkedList< cms::Connection * >** (p. 1272), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 802), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 802), **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 817), and **decaf::util::AbstractQueue< E >** (p. 110).

6.129.3.3 template<typename E> virtual void decaf::util::Collection< E >::clear ( ) [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

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

Implemented in **decaf::util::AbstractList< E >** (p. 100), **decaf::util::AbstractList< Pointer< Transport > >** (p. 100), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 100), **decaf::util::AbstractList< CompositeTask * >** (p. 100), **decaf::util::AbstractList< URI >** (p. 100), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 100), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 100), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 100), **decaf::util::AbstractList< Pointer< Command > >**

(p. 100), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 100), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 100), **decaf::util::AbstractList< cms::Destination * >** (p. 100), **decaf::util::AbstractList< cms::Session * >** (p. 100), **decaf::util::AbstractList< cms::Connection * >** (p. 100), **decaf::util::StlList< E >** (p. 1972), **decaf::util::PriorityQueue< E >** (p. 1679), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2059), **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1260), **decaf::util::LinkedList< E >** (p. 1274), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1274), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1274), **decaf::util::LinkedList< CompositeTask * >** (p. 1274), **decaf::util::LinkedList< URI >** (p. 1274), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1274), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1274), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1274), **decaf::util::LinkedList< Pointer< Command > >** (p. 1274), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1274), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1274), **decaf::util::LinkedList< cms::Destination * >** (p. 1274), **decaf::util::LinkedList< cms::Session * >** (p. 1274), **decaf::util::LinkedList< cms::Connection * >** (p. 1274), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 803), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 803), **decaf::util::StlSet< E >** (p. 1998), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 1998), **decaf::util::StlSet< Resource * >** (p. 1998), **decaf::util::ArrayList< E >** (p. 350), **decaf::util::AbstractQueue< E >** (p. 110), **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 818), **decaf::util::AbstractCollection< E >** (p. 88), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 88), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 88), **decaf::util::AbstractCollection< Resource * >** (p. 88), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 88), **decaf::util::AbstractCollection< CompositeTask * >** (p. 88), **decaf::util::AbstractCollection< URI >** (p. 88), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 88), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 88), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 88), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 88), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 88), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 88), **decaf::util::AbstractCollection< cms::Destination * >** (p. 88), **decaf::util::AbstractCollection< cms::Session * >** (p. 88), and **decaf::util::AbstractCollection< cms::Connection * >** (p. 88).

6.129.3.4 template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & *value*) const [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 (value == NULL ? e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

Implemented in **decaf::util::StlList< E >** (p. 1972), **decaf::util::ArrayList< E >** (p. 350), **decaf::util::LinkedList< E >** (p. 1275), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1275), **decaf::util::LinkedList< CompositeTask * >** (p. 1275), **decaf::util::LinkedList< URI >** (p. 1275), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1275), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1275), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1275), **decaf::util::LinkedList< Pointer< Command > >** (p. 1275), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1275), **decaf::util::LinkedList< cms::Destination * >** (p. 1275), **decaf::util::LinkedList< cms::Session * >** (p. 1275), **decaf::util::LinkedList< cms::Connection * >** (p. 1275), **decaf::util::concurrent::CopyOnWriteArrayList< E >**

(p. 803), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 803), **decaf::util::StlSet**< **E** > (p. 1998), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 1998), **decaf::util::StlSet**< **Resource** * > (p. 1998), **decaf::util::AbstractCollection**< **E** > (p. 88), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 88), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 88), **decaf::util::AbstractCollection**< **Resource** * > (p. 88), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 88), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 88), **decaf::util::AbstractCollection**< **URI** > (p. 88), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 88), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 88), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 88), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 88), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 88), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 88), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 88), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 88), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 88), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 818).

Referenced by decaf::util::AbstractSet< Resource * >::removeAll(), decaf::util::AbstractCollection< cms::Connection * >::removeAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::AbstractCollection< cms::Connection * >::retainAll(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll().

6.129.3.5 template<typename E> virtual bool decaf::util::Collection< E >::containsAll (const Collection< E > & *collection*) const [pure virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	The **Collection** (p. 660) to compare to this one.

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

Implemented in **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2059), **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 804), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 804), **decaf::util::AbstractCollection**< **E** > (p. 89), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 89), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 89), **decaf::util::AbstractCollection**< **Resource** * > (p. 89), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 89), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 89), **decaf::util::AbstractCollection**< **URI** > (p. 89), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 89), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 89), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 89), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 89), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 89), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 89), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 89), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 89), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 89), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 818).

6.129.3.6 template<typename E> virtual void decaf::util::Collection< E >::copy (const Collection< E > & *collection*) [pure virtual]

Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).

Parameters

collection	The collection to mirror.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
IllegalStateException	if the elements cannot be added at this time due to insertion restrictions.

Implemented in **decaf::util::StlList< E >** (p. 1973), **decaf::util::LinkedList< E >** (p. 1275), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1275), **decaf::util::LinkedList< CompositeTask * >** (p. 1275), **decaf::util::LinkedList< URI >** (p. 1275), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1275), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1275), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1275), **decaf::util::LinkedList< Pointer< Command > >** (p. 1275), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1275), **decaf::util::LinkedList< cms::Destination * >** (p. 1275), **decaf::util::LinkedList< cms::Session * >** (p. 1275), **decaf::util::LinkedList< cms::Connection * >** (p. 1275), **decaf::util::StlSet< E >** (p. 1999), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 1999), **decaf::util::StlSet< Resource * >** (p. 1999), **decaf::util::AbstractCollection< E >** (p. 89), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 89), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 89), **decaf::util::AbstractCollection< Resource * >** (p. 89), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 89), **decaf::util::AbstractCollection< CompositeTask * >** (p. 89), **decaf::util::AbstractCollection< URI >** (p. 89), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 89), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 89), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 89), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 89), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 89), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 89), **decaf::util::AbstractCollection< cms::Destination * >** (p. 89), **decaf::util::AbstractCollection< cms::Session * >** (p. 89), **decaf::util::AbstractCollection< cms::Connection * >** (p. 89), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 804), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 804), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 819).

6.129.3.7 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::StlList< E >** (p. 1973), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 804), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 804), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2061), **decaf::util::StlSet< E >** (p. 1999), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 1999), **decaf::util::StlSet< Resource * >** (p. 1999), **decaf::util::AbstractCollection< E >** (p. 90), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 90), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 90), **decaf::util::AbstractCollection< Resource * >** (p. 90), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 90), **decaf::util::AbstractCollection< CompositeTask * >** (p. 90), **decaf::util::AbstractCollection< URI >** (p. 90), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 90), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 90), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 90), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 90), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 90), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 90), **decaf::util::AbstractCollection< cms::Destination * >** (p. 90), **decaf::util::AbstractCollection< cms::Session * >** (p. 90), **decaf::util::AbstractCollection< cms::Connection * >** (p. 90), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 819).

6.129.3.8 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::StlList< E >** (p. 1974), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 806), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 806), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2061), **decaf::util::LinkedList< E >** (p. 1278), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1278), **decaf::util::LinkedList< CompositeTask * >** (p. 1278), **decaf::util::LinkedList< URI >** (p. 1278), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1278), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1278), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1278), **decaf::util::LinkedList< Pointer< Command > >** (p. 1278), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1278), **decaf::util::LinkedList< cms::Destination * >** (p. 1278), **decaf::util::LinkedList< cms::Session * >** (p. 1278), **decaf::util::LinkedList< cms::Connection * >** (p. 1278), **decaf::util::StlSet< E >** (p. 1999), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 1999), **decaf::util::StlSet< Resource * >** (p. 1999), **decaf::util::AbstractCollection< E >** (p. 90), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 90), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 90), **decaf::util::AbstractCollection< Resource * >** (p. 90), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 90), **decaf::util::AbstractCollection< CompositeTask * >** (p. 90), **decaf::util::AbstractCollection< URI >** (p. 90), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 90), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 90), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 90), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 90), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 90), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 90), **decaf::util::AbstractCollection< cms::Destination * >** (p. 90), **decaf::util::AbstractCollection< cms::Session * >** (p. 90), **decaf::util::AbstractCollection< cms::Connection * >** (p. 90), **decaf::util::ArrayList< E >** (p. 352), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 820).

Referenced by decaf::util::AbstractList< cms::Connection * >::addAll(), decaf::util::StlList< E >::addAll(), decaf::util::concurrent::SynchronousQueue< E >::containsAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll().

6.129.3.9 template<typename E> virtual bool decaf::util::Collection< E >::remove (const E & *value*) [pure virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationExceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implemented in **decaf::util::StlList< E >** (p. 1976), **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1265), **decaf::util::PriorityQueue< E >** (p. 1682), **decaf::util::ArrayList< E >** (p. 353), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 809), **decaf::util::concurrent::CopyOnWriteArrayList<**

ServiceListener * > (p. 809), **decaf::util::AbstractCollection**< **E** > (p. 92), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 92), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 92), **decaf::util::AbstractCollection**< **Resource** * > (p. 92), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 92), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 92), **decaf::util::AbstractCollection**< **URI** > (p. 92), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 92), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 92), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 92), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 92), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 92), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 92), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 92), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 92), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 92), **decaf::util::StlSet**< **E** > (p. 2000), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 2000), **decaf::util::StlSet**< **Resource** * > (p. 2000), **decaf::util::LinkedList**< **E** > (p. 1283), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1283), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1283), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1283), **decaf::util::LinkedList**< **URI** > (p. 1283), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1283), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1283), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1283), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1283), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1283), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1283), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1283), **decaf::util::LinkedList**< **cms::Session** * > (p. 1283), **decaf::util::LinkedList**< **cms::Connection** * > (p. 1283), and **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 820).

6.129.3.10 template<typename E> virtual bool decaf::util::Collection< E >::removeAll (const Collection< E > & *collection*) [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. 660) whose elements are to be removed from this one.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 809), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 809), **decaf::util::AbstractCollection**< **E** > (p. 93), **decaf::util::AbstractCollection**< **Pointer**< **Transport** > > (p. 93), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 93), **decaf::util::AbstractCollection**< **Resource** * > (p. 93), **decaf::util::AbstractCollection**< **cms::MessageConsumer** * > (p. 93), **decaf::util::AbstractCollection**< **CompositeTask** * > (p. 93), **decaf::util::AbstractCollection**< **URI** > (p. 93), **decaf::util::AbstractCollection**< **Pointer**< **MessageDispatch** > > (p. 93), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 93), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 93), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 93), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 93), **decaf::util::AbstractCollection**< **cms::MessageProducer** * > (p. 93), **decaf::util::AbstractCollection**< **cms::Destination** * > (p. 93), **decaf::util::AbstractCollection**< **cms::Session** * > (p. 93), **decaf::util::AbstractCollection**< **cms::Connection** * > (p. 93), **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 821), **decaf::util::AbstractSet**< **E** > (p. 119), **decaf::util::AbstractSet**< **Pointer**< **Synchronization** > > (p. 119), and **decaf::util::AbstractSet**< **Resource** * > (p. 119).

6.129.3.11 template<typename E> virtual bool decaf::util::Collection< E >::retainAll (const Collection< E > & *collection*) [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. 660) whose elements are to be retained.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 810), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 810), **decaf::util::AbstractCollection< E >** (p. 93), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 93), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 93), **decaf::util::AbstractCollection< Resource * >** (p. 93), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 93), **decaf::util::AbstractCollection< CompositeTask * >** (p. 93), **decaf::util::AbstractCollection< URI >** (p. 93), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 93), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 93), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 93), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 93), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 93), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 93), **decaf::util::AbstractCollection< cms::Destination * >** (p. 93), **decaf::util::AbstractCollection< cms::Session * >** (p. 93), **decaf::util::AbstractCollection< cms::Connection * >** (p. 93), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 821).

6.129.3.12 template<typename E> virtual int 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::concurrent::CopyOnWriteArrayList< E >** (p. 811), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 811), **decaf::util::StlList< E >** (p. 1977), **decaf::util::PriorityQueue< E >** (p. 1682), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2064), **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1265), **decaf::util::LinkedList< E >** (p. 1286), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1286), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1286), **decaf::util::LinkedList< CompositeTask * >** (p. 1286), **decaf::util::LinkedList< URI >** (p. 1286), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1286), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1286), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1286), **decaf::util::LinkedList< Pointer< Command > >** (p. 1286), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1286), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1286), **decaf::util::LinkedList< cms::Destination * >** (p. 1286), **decaf::util::LinkedList< cms::Session * >** (p. 1286), **decaf::util::LinkedList< cms::Connection * >** (p. 1286), **decaf::util::StlSet< E >** (p. 2000), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2000), **decaf::util::StlSet< Resource * >** (p. 2000), **decaf::util::ArrayList< E >** (p. 354), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 822).

Referenced by decaf::util::AbstractList< cms::Connection * >::add(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::ArrayList< E >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::ArrayList< E >::ArrayList(), decaf::util::AbstractList< cms::Connection * >::clear(), decaf::util::concurrent::CopyOnWriteArraySet< E >::equals(), decaf::util::AbstractCollection< cms::Connection * >::equals(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals(), decaf::util::AbstractCollection< cms::Connection * >::isEmpty(), decaf::util::AbstractList< cms::Connection * >::lastIndexOf(), decaf::util::AbstractSet< Resource * >::removeAll(), and decaf::util::AbstractCollection< cms::Connection * >::toArray().

6.129.3.13 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 in the form of an STL vector.

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1266), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 811), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 811), **decaf::util::ArrayList< E >** (p. 355), **decaf::util::AbstractCollection< E >** (p. 94), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 94), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 94), **decaf::util::AbstractCollection< Resource * >** (p. 94), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 94), **decaf::util::AbstractCollection< CompositeTask * >** (p. 94), **decaf::util::AbstractCollection< URI >** (p. 94), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 94), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 94), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 94), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 94), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 94), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 94), **decaf::util::AbstractCollection< cms::Destination * >** (p. 94), **decaf::util::AbstractCollection< cms::Session * >** (p. 94), **decaf::util::AbstractCollection< cms::Connection * >** (p. 94), **decaf::util::LinkedList< E >** (p. 1286), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1286), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1286), **decaf::util::LinkedList< CompositeTask * >** (p. 1286), **decaf::util::LinkedList< URI >** (p. 1286), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1286), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1286), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1286), **decaf::util::LinkedList< Pointer< Command > >** (p. 1286), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1286), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1286), **decaf::util::LinkedList< cms::Destination * >** (p. 1286), **decaf::util::LinkedList< cms::Session * >** (p. 1286), **decaf::util::LinkedList< cms::Connection * >** (p. 1286), **decaf::util::concurrent::SynchronousQueue< E >** (p. 2064), and **decaf::util::concurrent::CopyOnWriteArraySet< E >** (p. 822).

Referenced by decaf::util::ArrayList< E >::addAll(), and decaf::util::StlList< E >::addAll().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Collection.h**

6.130 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. 671) Id of this **Message** (p. 1412).*
- virtual int **getCommandId** () **const** =0

  *Gets the **Command** (p. 671) Id of this **Message** (p. 1412).*
- virtual void **setResponseRequired** (**const** bool required)=0

  *Set if this **Message** (p. 1412) requires a **Response** (p. 1781).*
- virtual bool **isResponseRequired** () **const** =0

  *Is a **Response** (p. 1781) required for this **Command** (p. 671).*
- 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

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual bool **isConnectionControl** () **const** =0
- 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

6.130.1 Constructor & Destructor Documentation

6.130.1.1 virtual activemq::commands::Command::~Command ( ) [inline, virtual]

6.130.2 Member Function Documentation

6.130.2.1 virtual int activemq::commands::Command::getCommandId ( ) const [pure virtual]

Gets the **Command** (p. 671) Id of this **Message** (p. 1412).

Returns

Command (p. 671) Id

Implemented in **activemq::commands::BaseCommand** (p. 383).

6.130.2.2 virtual bool activemq::commands::Command::isBrokerInfo ( ) const [pure virtual]

Implemented in **activemq::commands::BrokerInfo** (p. 446), and **activemq::commands::BaseCommand** (p. 383).

6.130.2.3 virtual bool activemq::commands::Command::isConnectionControl ( ) const [pure virtual]

Implemented in **activemq::commands::ConnectionControl** (p. 730), and **activemq::commands::BaseCommand** (p. 383).

6.130.2.4 virtual bool activemq::commands::Command::isConnectionInfo ( ) const [pure virtual]

Implemented in **activemq::commands::ConnectionInfo** (p. 754), and **activemq::commands::BaseCommand** (p. 383).

6.130.2.5 virtual bool activemq::commands::Command::isConsumerInfo ( ) const [pure virtual]

Implemented in **activemq::commands::ConsumerInfo** (p. 786), and **activemq::commands::BaseCommand** (p. 383).

6.130.2.6 virtual bool activemq::commands::Command::isKeepAliveInfo ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 384), and **activemq::commands::KeepAliveInfo** (p. 1235).

6.130.2.7 virtual bool activemq::commands::Command::isMessage ( ) const [pure virtual]

Implemented in **activemq::commands::Message** (p. 1421), and **activemq::commands::BaseCommand** (p. 384).

6.130.2.8 virtual bool activemq::commands::Command::isMessageAck ( ) const [pure virtual]

Implemented in **activemq::commands::MessageAck** (p. 1451), and **activemq::commands::BaseCommand** (p. 384).

6.130.2.9 virtual bool activemq::commands::Command::isMessageDispatch ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 384), and **activemq::commands::MessageDispatch** (p. 1461).

6.130.2.10 virtual bool activemq::commands::Command::isMessageDispatchNotification ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 384), and **activemq::commands::MessageDispatchNotification** (p. 1471).

6.130.2.11 virtual bool activemq::commands::Command::isProducerAck ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 384), and **activemq::commands::ProducerAck** (p. 1685).

6.130.2.12 virtual bool activemq::commands::Command::isProducerInfo ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 384), and **activemq::commands::ProducerInfo** (p. 1700).

6.130.2.13 virtual bool activemq::commands::Command::isRemoveInfo ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 384), and **activemq::commands::RemoveInfo** (p. 1760).

6.130.2.14 virtual bool activemq::commands::Command::isRemoveSubscriptionInfo ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 385), and **activemq::commands::RemoveSubscriptionInfo** (p. 1766).

6.130.2.15 virtual bool activemq::commands::Command::isResponse ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 385), and **activemq::commands::Response** (p. 1783).

6.130.2.16 virtual bool activemq::commands::Command::isResponseRequired ( ) const [pure virtual]

Is a **Response** (p. 1781) required for this **Command** (p. 671).

Returns

true if a response is required.

Implemented in **activemq::commands::BaseCommand** (p. 385).

6.130.2.17 virtual bool activemq::commands::Command::isShutdownInfo ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 385), and **activemq::commands::ShutdownInfo** (p. 1885).

6.130.2.18 virtual bool activemq::commands::Command::isTransactionInfo ( ) const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 385), and **activemq::commands::TransactionInfo** (p. 2151).

6.130.2.19 virtual bool activemq::commands::Command::isWireFormatInfo ( ) const [pure virtual]

Implemented in **activemq::commands::WireFormatInfo** (p. 2245), and **activemq::commands::BaseCommand** (p. 385).

6.130.2.20 virtual void activemq::commands::Command::setCommandId (int *id*) [pure virtual]

Sets the **Command** (p. 671) Id of this **Message** (p. 1412).

Parameters

id	**Command** (p. 671) Id

Implemented in **activemq::commands::BaseCommand** (p. 385).

6.130.2.21 virtual void activemq::commands::Command::setResponseRequired (const bool *required*) [pure virtual]

Set if this **Message** (p. 1412) requires a **Response** (p. 1781).

Parameters

required	true if response is required

Implemented in **activemq::commands::BaseCommand** (p. 386).

6.130.2.22 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. 411).

Implemented in **activemq::commands::Message** (p. 1424), **activemq::commands::ConsumerInfo** (p. 788), **activemq::commands::ActiveMQBytesMessage** (p. 138), **activemq::commands::BrokerInfo** (p. 447), **activemq::commands::ConnectionInfo** (p. 755), **activemq::commands::MessageAck** (p. 1451), **activemq::commands::ConsumerControl** (p. 772), **activemq::commands::ProducerInfo** (p. 1700), **activemq::commands::ConnectionControl** (p. 731), **activemq::commands::DestinationInfo** (p. 942), **activemq::commands::MessagePull** (p. 1505), **activemq::commands::SessionInfo** (p. 1851), **activemq::commands::MessageDispatch** (p. 1461), **activemq::commands::MessageDispatchNotification** (p. 1472), **activemq::commands::TransactionInfo** (p. 2151), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::RemoveSubscriptionInfo** (p. 1766), **activemq::commands::ActiveMQStreamMessage** (p. 286), **activemq::commands::ProducerAck** (p. 1685), **activemq::commands::RemoveInfo** (p. 1760), **activemq::commands::ActiveMQMapMessage** (p. 219), **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 865), **activemq::commands::ExceptionResponse** (p. 998), **activemq::commands::ReplayCommand** (p. 1772), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::Response** (p. 1783), **activemq::commands::FlushCommand** (p. 1070), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::BaseCommand** (p. 386), **activemq::commands::ActiveMQBlobMessage** (p. 125), **activemq::commands::WireFormatInfo** (p. 2247), **activemq::commands::ActiveMQTextMessage** (p. 317), **activemq::commands::ActiveMQObjectMessage** (p. 235), and **activemq::commands::ActiveMQMessage** (p. 225).

6.130.2.23 virtual decaf::lang::Pointer<commands::Command> activemq::commands::Command::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implemented in **activemq::commands::Message** (p. 1424), **activemq::commands::ConsumerInfo** (p. 788), **activemq::commands::BrokerInfo** (p. 447), **activemq::commands::ConnectionInfo** (p. 755), **activemq::commands::MessageAck** (p. 1451), **activemq::commands::ConnectionControl** (p. 731), **activemq::commands::ProducerInfo** (p. 1701), **activemq::commands::ConsumerControl** (p. 772), **activemq::commands::MessageDispatch** (p. 1462), **activemq::commands::MessageDispatchNotification** (p. 1472), **activemq::commands::MessagePull** (p. 1505), **activemq::commands::DestinationInfo** (p. 942), **activemq::commands::RemoveSubscriptionInfo** (p. 1767), **activemq::commands::TransactionInfo** (p. 2151), **activemq::commands::BrokerError** (p. 436), **activemq::commands::SessionInfo** (p. 1851), **activemq::commands::ProducerAck** (p. 1685), **activemq::commands::RemoveInfo** (p. 1760), **activemq::commands::ConnectionError** (p. 738), **activemq::commands::Response** (p. 1784), **activemq::commands::Replay-**

Command (p. 1772), **activemq::commands::KeepAliveInfo** (p. 1236), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::ControlCommand** (p. 795), **activemq::commands::FlushCommand** (p. 1070), and **activemq::commands::WireFormatInfo** (p. 2248).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Command.h**

6.131 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
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveInfo** (**commands::RemoveInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionInfo** (**commands::ConnectionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processSessionInfo** (**commands::SessionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerInfo** (**commands::ProducerInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerInfo** (**commands::ConsumerInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveConnection** (**commands::ConnectionId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSession** (**commands::SessionId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveProducer** (**commands::ProducerId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveConsumer** (**commands::ConsumerId** *id)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processDestinationInfo** (**commands::DestinationInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveDestination** (**commands::DestinationInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessage** (**commands::Message** *send)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageAck** (**commands::MessageAck** *ack)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessagePull** (**commands::MessagePull** *pull)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBeginTransaction** (**commands::TransactionInfo** *info)=0

- virtual **decaf::lang::Pointer** < **commands::Command** > **processPrepareTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerError** (**commands::BrokerError** *error)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** *replay)=0
- virtual **decaf::lang::Pointer** < **commands::Command** > **processResponse** (**commands::Response** *response)=0

6.131.1 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

6.131.2 Constructor & Destructor Documentation

6.131.2.1 virtual activemq::state::CommandVisitor::~CommandVisitor ( ) [inline, virtual]

6.131.3 Member Function Documentation

6.131.3.1 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBeginTransaction (commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 765).

6.131.3.2 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBrokerError (commands::BrokerError * *error*) [pure virtual]

6.131.3.3 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBrokerInfo (commands::BrokerInfo * *info*) [pure virtual]

6.131.3.4 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processCommitTransactionOnePhase (commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.5 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processCommitTransactionTwoPhase (commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.6 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConnectionControl (commands::ConnectionControl * *control*) [pure virtual]

6.131.3.7 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConnectionError (commands::ConnectionError * *error*) [pure virtual]

6.131.3.8 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConnectionInfo (commands::ConnectionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.9 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConsumerControl (commands::ConsumerControl * *control*) [pure virtual]

6.131.3.10 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConsumerInfo (commands::ConsumerInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.11 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processControlCommand (commands::ControlCommand * *command*) [pure virtual]

6.131.3.12 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processDestinationInfo (commands::DestinationInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.13 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processEndTransaction (commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.14 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processFlushCommand (commands::FlushCommand * *command*) [pure virtual]

6.131.3.15 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processForgetTransaction (commands::TransactionInfo * *info*) [pure virtual]

6.131.3.16 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processKeepAliveInfo (commands::KeepAliveInfo * *info*) [pure virtual]

6.131.3.17 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessage (commands::Message * *send*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.18 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessageAck (commands::MessageAck * *ack*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.19 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessageDispatch (commands::MessageDispatch * *dispatch*) [pure virtual]

6.131.3.20 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessageDispatchNotification (commands::MessageDispatchNotification * *notification*) [pure virtual]

6.131.3.21 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessagePull (commands::MessagePull * *pull*) [pure virtual]

6.131.3.22 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processPrepareTransaction (commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.23 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processProducerAck (commands::ProducerAck * *ack*) [pure virtual]

6.131.3.24 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processProducerInfo (commands::ProducerInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 766).

6.131.3.25 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRecoverTransactions (commands::TransactionInfo * *info*) [pure virtual]

6.131.3.26 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveConnection (commands::ConnectionId * *id*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 767).

6.131.3.27 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveConsumer (commands::ConsumerId * *id*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 767).

6.131.3.28 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveDestination (commands::DestinationInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 767).

6.131.3.29 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveInfo (commands::RemoveInfo * *info*) [pure virtual]

Implemented in **activemq::state::CommandVisitorAdapter** (p. 685).

6.131.3.30 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveProducer (commands::ProducerId * *id*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 767).

6.131.3.31 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveSession (commands::SessionId * *id*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 767).

6.131.3.32 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo * *info*) [pure virtual]

6.131.3.33 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processReplayCommand (commands::ReplayCommand * *replay*) [pure virtual]

6.131.3.34 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processResponse (commands::Response * *response*) [pure virtual]

6.131.3.35 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRollbackTransaction (commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 767).

6.131.3.36 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processSessionInfo (commands::SessionInfo * *info*) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 767).

6.131.3.37 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processShutdownInfo (commands::ShutdownInfo * *info*) [pure virtual]

6.131.3.38 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processTransactionInfo (commands::TransactionInfo * *info*) [pure virtual]

Implemented in **activemq::state::CommandVisitorAdapter** (p. 686).

6.131.3.39 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processWireFormat (commands::WireFormatInfo * *info*) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/state/**CommandVisitor.h**

6.132 activemq::state::CommandVisitorAdapter Class Reference

Default Implementation of a **CommandVisitor** (p. 675) 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)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSession** (**commands::SessionId** *id AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveProducer** (**commands::ProducerId** *id AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveConsumer** (**commands::ConsumerId** *id AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processDestinationInfo** (**commands::DestinationInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveDestination** (**commands::DestinationInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessage** (**commands::Message** *send AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageAck** (**commands::MessageAck** *ack AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessagePull** (**commands::MessagePull** *pull AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBeginTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processPrepareTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info AMQCPP_UNUSED)

- virtual **decaf::lang::Pointer** < **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processBrokerError** (**commands::BrokerError** *error AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** *replay AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processResponse** (**commands::Response** *response AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConnectionInfo** (**commands::ConnectionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processSessionInfo** (**commands::SessionInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processProducerInfo** (**commands::ProducerInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processConsumerInfo** (**commands::ConsumerInfo** *info AMQCPP_UNUSED)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processTransactionInfo** (**commands::TransactionInfo** *info)
- virtual **decaf::lang::Pointer** < **commands::Command** > **processRemoveInfo** (**commands::RemoveInfo** *info)

6.132.1 Detailed Description

Default Implementation of a **CommandVisitor** (p. 675) that returns NULL for all calls.

Since

3.0

6.132.2 Constructor & Destructor Documentation

6.132.2.1 virtual **activemq::state::CommandVisitorAdapter::~CommandVisitorAdapter** ( ) [inline, virtual]

6.132.3 Member Function Documentation

6.132.3.1 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processBeginTransaction** (**commands::TransactionInfo** *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.2 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processBrokerError** (**commands::BrokerError** *error *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.3 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processBrokerInfo** (**commands::BrokerInfo** *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.4 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase** (**commands::TransactionInfo** *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.5 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.6 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConnectionControl** (**commands::ConnectionControl** *control *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.7 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConnectionError** (**commands::ConnectionError** *error *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.8 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConnectionInfo** (**commands::ConnectionInfo** *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.9 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConsumerControl** (**commands::ConsumerControl** *control *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.10 virtual **decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConsumerInfo** (**commands::ConsumerInfo** *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.11 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processControlCommand (commands::ControlCommand *command *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.12 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processDestinationInfo (commands::DestinationInfo *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.13 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processEndTransaction (commands::TransactionInfo *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.14 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processFlushCommand (commands::FlushCommand *command *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.15 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processForgetTransaction (commands::TransactionInfo *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.16 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processKeepAliveInfo (commands::KeepAliveInfo *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.17 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processMessage (commands::Message *send *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.18 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processMessageAck (commands::MessageAck *ack *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.19 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processMessageDispatch (commands::MessageDispatch *dispatch *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.20 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processMessageDispatchNotification (commands::MessageDispatchNotification *notification *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.21 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processMessagePull (commands::MessagePull *pull *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.22 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processPrepareTransaction (commands::TransactionInfo *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.23 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processProducerAck (commands::ProducerAck *ack *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.24 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processProducerInfo (commands::ProducerInfo *info *AMQCPP_UNUSED*) [inline, virtual]

6.132.3.25 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRecoverTransactions (commands::TransactionInfo *info AMQCPP_UNUSED) [inline, virtual]

6.132.3.26 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveConnection (commands::ConnectionId *id AMQCPP_UNUSED) [inline, virtual]

6.132.3.27 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveConsumer (commands::ConsumerId *id AMQCPP_UNUSED) [inline, virtual]

6.132.3.28 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveDestination (commands::DestinationInfo *info AMQCPP_UNUSED) [inline, virtual]

6.132.3.29 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveInfo (commands::RemoveInfo * info) [inline, virtual]

Implements **activemq::state::CommandVisitor** (p. 680).

References activemq::commands::RemoveInfo::getObjectId(), activemq::commands::ConnectionId::ID_CONNECTIONID, activemq::commands::ConsumerId::ID_CONSUMERID, activemq::commands::ProducerId::ID_PRODUCERID, and activemq::commands::SessionId::ID_SESSIONID.

6.132.3.30 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveProducer (commands::ProducerId *id AMQCPP_UNUSED) [inline, virtual]

6.132.3.31 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveSession (commands::SessionId *id AMQCPP_UNUSED) [inline, virtual]

6.132.3.32 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED) [inline, virtual]

6.132.3.33 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processReplayCommand (commands::ReplayCommand *replay AMQCPP_UNUSED) [inline, virtual]

6.132.3.34 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processResponse (commands::Response *response AMQCPP_UNUSED) [inline, virtual]

6.132.3.35 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRollbackTransaction (commands::TransactionInfo *info AMQCPP_UNUSED) [inline, virtual]

6.132.3.36 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processSessionInfo (commands::SessionInfo *info AMQCPP_UNUSED) [inline, virtual]

6.132.3.37 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processShutdownInfo (commands::ShutdownInfo *info AMQCPP_UNUSED) [inline, virtual]

6.132.3.38 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processTransactionInfo (commands::TransactionInfo * *info*) [inline, virtual]

Implements **activemq::state::CommandVisitor** (p. 681).

References activemq::commands::TransactionInfo::getType(), 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.

6.132.3.39 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processWireFormat (commands::WireFormatInfo *info *AMQCPP_UNUSED*) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/state/**CommandVisitorAdapter.h**

6.133 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.

6.133.1 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.

6.133.2 Constructor & Destructor Documentation

6.133.2.1 template<typename T> virtual decaf::lang::Comparable< T >::~Comparable ( ) [inline, virtual]

6.133.3 Member Function Documentation

6.133.3.1 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 implementer 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. 686) 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::Double** (p. 959), **decaf::lang::Float** (p. 1043), **decaf::lang::Integer** (p. 1165), **decaf::lang::Long** (p. 1341), **decaf::lang::Boolean** (p. 424), **decaf::lang::Short** (p. 1860), **decaf::lang::Byte** (p. 478), **decaf::lang::Character** (p. 595), **decaf::lang::Double** (p. 959), **decaf::lang::Float** (p. 1043), **decaf::lang::Boolean** (p. 423), **decaf::lang::Integer** (p. 1164), **decaf::lang::Long** (p. 1341), **decaf::lang::Byte** (p. 478), **decaf::lang::Short** (p. 1860), and **decaf::lang::Character** (p. 595).

6.133.3.2 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::Byte** (p. 479), **decaf::lang::Boolean** (p. 424), **decaf::lang::Character** (p. 596), **decaf::lang::Byte** (p. 479), **decaf::lang::Double** (p. 961), **decaf::lang::Float** (p. 1044), **decaf::lang::Character** (p. 596), **decaf::lang::Integer** (p. 1166), **decaf::lang::Long** (p. 1342), **decaf::lang::Short** (p. 1861), **decaf::lang::Boolean** (p. 424), **decaf::lang::Double** (p. 961), **decaf::lang::Float** (p. 1043), **decaf::lang::Integer** (p. 1166), **decaf::lang::Long** (p. 1342), and **decaf::lang::Short** (p. 1861).

6.133.3.3 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::Double** (p. 963), **decaf::lang::Float** (p. 1047), **decaf::lang::Integer** (p. 1168), **decaf::lang::Long** (p. 1345), **decaf::lang::Short** (p. 1862), **decaf::lang::Boolean** (p. 424), **decaf::lang::Byte** (p. 480), **decaf::lang::Character** (p. 598), **decaf::lang::Double** (p. 962), **decaf::lang::Float** (p. 1046), **decaf::lang::Integer** (p. 1168), **decaf::lang::Long** (p. 1345), **decaf::lang::Short** (p. 1862), **decaf::lang::Boolean** (p. 424), **decaf::lang::Byte** (p. 480), and **decaf::lang::Character** (p. 598).

6.133.3.4 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::Double** (p. 963), **decaf::lang::Float** (p. 1047), **decaf::lang::Integer** (p. 1169), **decaf::lang::Long** (p. 1345), **decaf::lang::Short** (p. 1863), **decaf::lang::Boolean** (p. 425), **decaf::lang::Byte** (p. 481), **decaf::lang::Character** (p. 598), **decaf::lang::Double** (p. 963), **decaf::lang::Float** (p. 1047), **decaf::lang::Integer** (p. 1168), **decaf::lang::Long** (p. 1345), **decaf::lang::Boolean** (p. 425), **decaf::lang::Short** (p. 1862), **decaf::lang::Byte** (p. 481), and **decaf::lang::Character** (p. 598).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Comparable.h**

6.134 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. 689) to be passed to an STL **Map** (p. 1371) for use as the sorting criteria.*
- virtual int **compare** (**const** T &o1, **const** T &o2) **const** =0

  Compares its two arguments for order.

6.134.1 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. 689) 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.

Since

1.0

6.134.2 Constructor & Destructor Documentation

6.134.2.1 template<typename T> virtual decaf::util::Comparator< T >::~Comparator ( ) [inline, virtual]

6.134.3 Member Function Documentation

6.134.3.1 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. 1251).

6.134.3.2 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. 689) to be passed to an STL **Map** (p. 1371) 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. 1251).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Comparator.h**

6.135 activemq::util::CompositeData Class Reference

Represents a Composite URI.

`#include <src/main/activemq/util/CompositeData.h>`

Public Member Functions

- **CompositeData** ()
- virtual ~**CompositeData** ()
- **LinkedList**< **URI** > & **getComponents** ()
- **const LinkedList**< **URI** > & **getComponents** () **const**
- void **setComponents** (**const LinkedList**< **URI** > &components)
- std::string **getFragment** () **const**
- void **setFragment** (**const** std::string &fragment)
- **const Properties** & **getParameters** () **const**
- void **setParameters** (**const Properties** &parameters)
- 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**

6.135.1 Detailed Description

Represents a Composite URI.

Since

3.0

6.135.2 Constructor & Destructor Documentation

6.135.2.1 activemq::util::CompositeData::CompositeData ( )

6.135.2.2 virtual activemq::util::CompositeData::~CompositeData ( ) [virtual]

6.135.3 Member Function Documentation

6.135.3.1 LinkedList<URI>& activemq::util::CompositeData::getComponents ( ) [inline]

6.135.3.2 const LinkedList<URI>& activemq::util::CompositeData::getComponents ( ) const [inline]

6.135.3.3 std::string activemq::util::CompositeData::getFragment ( ) const [inline]

6.135.3.4 std::string activemq::util::CompositeData::getHost ( ) const [inline]

6.135.3.5 const Properties& activemq::util::CompositeData::getParameters ( ) const [inline]

6.135.3.6 std::string activemq::util::CompositeData::getPath ( ) const [inline]

6.135.3.7 std::string activemq::util::CompositeData::getScheme ( ) const [inline]

6.135.3.8 void activemq::util::CompositeData::setComponents (const LinkedList< URI > & *components*) [inline]

6.135.3.9 void activemq::util::CompositeData::setFragment (const std::string & *fragment*) [inline]

6.135.3.10 void activemq::util::CompositeData::setHost (const std::string & *host*) [inline]

6.135.3.11 void activemq::util::CompositeData::setParameters (const Properties & *parameters*) [inline]

6.135.3.12 void activemq::util::CompositeData::setPath (const std::string & *path*) [inline]

6.135.3.13 void activemq::util::CompositeData::setScheme (const std::string & *scheme*) [inline]

6.135.3.14 URI activemq::util::CompositeData::toURI ( ) const

Exceptions

decaf::net::URISyntax-Exception *(p. 2214)*	

The documentation for this class was generated from the following file:

- src/main/activemq/util/**CompositeData.h**

6.136 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. 693).

`#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. 2073) in the **CompositeTaskRunner** (p. 693)'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`.

6.136.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 693).

Since

3.0

6.136.2 Constructor & Destructor Documentation

6.136.2.1 virtual activemq::threads::CompositeTask::~CompositeTask ( ) [inline, virtual]

6.136.3 Member Function Documentation

6.136.3.1 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. 2073) in the **CompositeTaskRunner** (p. 693)'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::FailoverTransport** (p. 1015), **activemq::transport::failover::BackupTransportPool** (p. 380), and **activemq::transport::failover::CloseTransportsTask** (p. 635).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTask.h**

6.137 activemq::threads::CompositeTaskRunner Class Reference

A **Task** (p. 2073) 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. 692) to the Set of Tasks that this class manages.*
- void **removeTask** (**CompositeTask** *task)

  *Removes a **CompositeTask** (p. 692) 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** (p. 2074)'s thread has exited.*
- virtual void **wakeup** ()

  *Signal the **TaskRunner** (p. 2074) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2073) 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** ()

  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.

6.137.1 Detailed Description

A **Task** (p. 2073) 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

6.137.2 Constructor & Destructor Documentation

6.137.2.1 activemq::threads::CompositeTaskRunner::CompositeTaskRunner ( )

6.137.2.2 virtual activemq::threads::CompositeTaskRunner::~CompositeTaskRunner ( ) [virtual]

6.137.3 Member Function Documentation

6.137.3.1 void activemq::threads::CompositeTaskRunner::addTask (CompositeTask * *task*)

Adds a new **CompositeTask** (p. 692) to the Set of Tasks that this class manages.

Parameters

task	- Pointer to a **CompositeTask** (p. 692) instance.

6.137.3.2 virtual bool activemq::threads::CompositeTaskRunner::iterate ( ) [protected, 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.

Implements **activemq::threads::Task** (p. 2073).

6.137.3.3 void activemq::threads::CompositeTaskRunner::removeTask (CompositeTask * *task*)

Removes a **CompositeTask** (p. 692) that was added previously.

Parameters

task	- Pointer to a **CompositeTask** (p. 692) instance.

6.137.3.4 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. 1793).

6.137.3.5 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.

Implements **activemq::threads::TaskRunner** (p. 2074).

6.137.3.6 virtual void activemq::threads::CompositeTaskRunner::shutdown ( ) [virtual]

Shutdown once the task has finished and the **TaskRunner** (p. 2074)'s thread has exited.

Implements **activemq::threads::TaskRunner** (p. 2074).

6.137.3.7 virtual void activemq::threads::CompositeTaskRunner::wakeup ( ) [virtual]

Signal the **TaskRunner** (p. 2074) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2073) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 2074).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTaskRunner.h**

6.138 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 2161) is a **Transport** (p. 2161) 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** (bool rebalance, **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. 2161) is a composite of.*
- virtual void **removeURI** (bool rebalance, **const List**< **URI** > &uris)=0

  *Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2161) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2161) should result in that **Transport** (p. 2161) being disposed of.*

6.138.1 Detailed Description

A Composite **Transport** (p. 2161) is a **Transport** (p. 2161) implementation that is composed of several Transports.

The composition could be such that only one **Transport** (p. 2161) exists for each URI that is composed or there could be many active Transports working at once.

Since

3.0

6.138.2 Constructor & Destructor Documentation

6.138.2.1 virtual activemq::transport::CompositeTransport::~CompositeTransport ( ) [inline, virtual]

6.138.3 Member Function Documentation

6.138.3.1 virtual void activemq::transport::CompositeTransport::addURI (bool *rebalance,* 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. 2161) is a composite of.

Parameters

rebalance	Indicates if the addition should cause a forced reconnect or not.
uris	The new URI set to add to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1012).

6.138.3.2 virtual void activemq::transport::CompositeTransport::removeURI (bool *rebalance,* 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. 2161) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2161) should result in that **Transport** (p. 2161) being disposed of.

Parameters

rebalance	Indicates if the removal should cause a forced reconnect or not.
uris	The new URI set to remove to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1017).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**CompositeTransport.h**

6.139 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 1371) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1371) 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

  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

  Replace entry for key only if currently mapped to some value.

6.139.1 Detailed Description

template<typename K, typename V, typename COMPARATOR>class decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >

Interface for a **Map** (p. 1371) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1371) interface.

Since

1.0

6.139.2 Constructor & Destructor Documentation

6.139.2.1 template<typename K, typename V, typename COMPARATOR> virtual decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::~ConcurrentMap ( ) [inline, virtual]

6.139.3 Member Function Documentation

6.139.3.1 template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::putIfAbsent (const K & *key*, const V & *value*) [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

UnsupportedOperation-Exception	if the put operation is not supported by this map

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 710), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 710), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 710), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 710), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 710), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 710), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 710).

6.139.3.2 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. 711), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 711), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 711), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 711), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 711), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 711), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 711).

6.139.3.3 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. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 712), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 712).

6.139.3.4 template<typename K, typename V, typename COMPARATOR> virtual V decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (const K & *key,* const V & *value*) [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 (p. 1537)(...);
};
```

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** (p. 1537) if there was no mapping for key.

Exceptions

NoSuchElementException *(p. 1537)*	if there was no previous mapping.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 712), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 712), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 712).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentMap.h**

6.140 decaf::util::ConcurrentModificationException Class Reference

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

Inheritance diagram for decaf::util::ConcurrentModificationException:

Public Member Functions

- **ConcurrentModificationException** () throw ()

  Default Constructor.
- **ConcurrentModificationException** (**const lang::Exception** &ex) throw ()

  Copy Constructor.
- **ConcurrentModificationException** (**const ConcurrentModificationException** &ex) throw ()

Copy Constructor.

- **ConcurrentModificationException** (**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.
- **ConcurrentModificationException** (**const** std::exception ∗**cause**) throw ()

  Constructor.
- **ConcurrentModificationException** (**const** char ∗file, **const** int lineNumber, **const** char ∗msg,...) throw ()

  Constructor.
- virtual
  ConcurrentModificationException ∗ **clone** () **const**

  Clones this exception.
- virtual ∼**ConcurrentModificationException** () throw ()

6.140.1 Constructor & Destructor Documentation

6.140.1.1 decaf::util::ConcurrentModificationException::ConcurrentModificationException ( ) throw ()

Default Constructor.

6.140.1.2 decaf::util::ConcurrentModificationException::ConcurrentModificationException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.140.1.3 decaf::util::ConcurrentModificationException::ConcurrentModificationException (const ConcurrentModificationException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.140.1.4 decaf::util::ConcurrentModificationException::ConcurrentModificationException (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

6.140.1.5 decaf::util::ConcurrentModificationException::ConcurrentModificationException (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.

6.140.1.6 decaf::util::ConcurrentModificationException::ConcurrentModificationException (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

6.140.1.7 virtual decaf::util::ConcurrentModificationException::~ConcurrentModificationException ( ) throw () [virtual]

6.140.2 Member Function Documentation

6.140.2.1 virtual ConcurrentModificationException* decaf::util::ConcurrentModificationException::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::RuntimeException** (p. 1796).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**ConcurrentModificationException.h**

6.141 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 1371) 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

UnsupportedOperation-Exception	*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. 1371) contains any element or not, TRUE or FALSE*

- virtual int **size** () **const**

  Returns

  The number of elements (key/value pairs) in this map.

- virtual V & **get** (**const** K &key)

  *Gets the value mapped to the specified key in the **Map** (p. 1371).*
  *If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.*

  Parameters

key	*The search key.*

  Returns

  A reference to the value for the given key.

  Exceptions

NoSuchElementException (p. 1537)	*if the key requests doesn't exist in the **Map** (p. 1371).*

- virtual **const** V & **get** (**const** K &key) **const**

  *Gets the value mapped to the specified key in the **Map** (p. 1371).*
  *If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.*

  Parameters

key	*The search key.*

  Returns

  A {const} reference to the value for the given key.

  Exceptions

NoSuchElementException (p. 1537)	*if the key requests doesn't exist in the **Map** (p. 1371).*

- virtual void **put** (**const** K &key, **const** V &value)

  Sets the value for the specified key.

  Parameters

key	*The target key.*
value	*The value to be set.*

  Exceptions

UnsupportedOperation-Exception	*if this map is unmodifiable.*

- virtual void **putAll** (**const ConcurrentStlMap**< K, V, COMPARATOR > &other)

- virtual void **putAll** (**const Map**< K, V, COMPARATOR > &other)

  *Stores a copy of the Mappings contained in the other **Map** (p. 1371) in this one.*

  Parameters

other	*A **Map** (p. 1371) instance whose elements are to all be inserted in this **Map** (p. 1371).*

  Exceptions

UnsupportedOperation-Exception	*If the implementing class does not support the putAll operation.*

- virtual V **remove** (**const** K &key)

  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 (p. 1537)	*if this key is not in the **Map** (p. 1371).*
UnsupportedOperation-Exception	*if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () **const**

  *Returns a **Set** (p. 1857) 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. 1210), **Set.remove** (p. 667), 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)

  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)

  Replace entry for key only if currently mapped to some value.

- virtual void **lock** ()

  Locks the object.

- virtual bool **tryLock** ()

  *Attempts to **Lock** (p. 1304) the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** ()

  Unlocks the object.

- virtual void **wait** ()

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

- virtual void **wait** (long long millisecs)

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

- virtual void **wait** (long long millisecs, int nanos)

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

- virtual void **notify** ()

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

- virtual void **notifyAll** ()

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

6.141.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>>class decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >

Map (p. 1371) 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. 1371) extends the **ConcurrentMap** (p. 696) inteface 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

6.141.2 Constructor & Destructor Documentation

6.141.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::ConcurrentStlMap ( ) [inline]

Default constructor - does nothing.

6.141.2.2 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.

6.141.2.3 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.

6.141.2.4 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::~ConcurrentStlMap ( ) [inline, virtual]

6.141.3 Member Function Documentation

6.141.3.1 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. 1373).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy().

6.141.3.2 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. 1373).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, 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().

6.141.3.3 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. 1374).

6.141.3.4 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< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap().

6.141.3.5 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. 1374).

6.141.3.6 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]

6.141.3.7 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. 1371) to compare to this one.

Returns

true if the **Map** (p. 1371) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1374).

6.141.3.8 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::get (const K & *key*) [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1371).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.

Parameters

key	The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException *(p. 1537)*	if the key requests doesn't exist in the **Map** (p. 1371).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1375).

6.141.3.9 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 [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1371).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.

Parameters

key	The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException *(p. 1537)*	if the key requests doesn't exist in the **Map** (p. 1371).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1375).

6.141.3.10 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. 1371) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1376).

6.141.3.11 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. 1857) 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. 1210), **Set.remove** (p. 667), 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. 1376).

6.141.3.12 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::lock ( ) [inline, virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

6.141.3.13 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notify ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.
RuntimeException	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2047).

6.141.3.14 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notifyAll ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.
RuntimeException	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2048).

6.141.3.15 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*) [inline, virtual]

Sets the value for the specified key.

Parameters

key	The target key.
value	The value to be set.

Exceptions

UnsupportedOperation-Exception	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1377).

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().

6.141.3.16 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*) [inline, virtual]

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy().

6.141.3.17 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*) [inline, virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 1371) in this one.

Parameters

other	A **Map** (p. 1371) instance whose elements are to all be inserted in this **Map** (p. 1371).

Exceptions

UnsupportedOperation-Exception	If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1377).

6.141.3.18 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*) [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

UnsupportedOperation-Exception	if the put operation is not supported by this map

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 697).

6.141.3.19 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & *key*) [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 *(p. 1537)*	if this key is not in the **Map** (p. 1371).
UnsupportedOperation-Exception	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1378).

6.141.3.20 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. 697).

6.141.3.21 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. 698).

6.141.3.22 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*) [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 (p. 1537)(...);
};
```

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** (p. 1537) if there was no mapping for key.

Exceptions

NoSuchElementException *(p. 1537)*	if there was no previous mapping.

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 699).

6.141.3.23 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual int 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. 1378).

6.141.3.24 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::tryLock ( ) [inline, virtual]

Attempts to **Lock** (p. 1304) 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. 2048).

6.141.3.25 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::unlock ( ) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

6.141.3.26 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. 1379).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values().

6.141.3.27 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.141.3.28 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.141.3.29 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*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentStlMap.h**

6.142 decaf::util::concurrent::locks::Condition Class Reference

Condition (p. 715) factors out the **Mutex** (p. 1519) 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. 1305) implementations.

`#include <src/main/decaf/util/concurrent/locks/Condition.h>`

Public Member Functions

- virtual ~**Condition** ()
- virtual void **await** ()=0

  Causes the current thread to wait until it is signaled or interrupted.
- virtual void **awaitUninterruptibly** ()=0

  Causes the current thread to wait until it is signalled.
- virtual long long **awaitNanos** (long long nanosTimeout)=0

  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

  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
- virtual void **signal** ()=0

  Wakes up one waiting thread.
- virtual void **signalAll** ()=0

  Wakes up all waiting threads.

6.142.1 Detailed Description

Condition (p. 715) factors out the **Mutex** (p. 1519) 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. 1305) implementations.

Where a **Lock** (p. 1305) replaces the use of synchronized statements, a **Condition** (p. 715) 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. 715) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 715) instance for a particular **Lock** (p. 1305) 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. 715) 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. 717); items[putptr] = x; if (++putptr == 100) putptr = 0; ++count; notEmpty->**signal()** (p. 719); } catch(...) { lock->unlock(); } }

public Object take() throw(InterruptedException) { lock->lock(); try { while(count == 0) notEmpty->**await()** (p. 717); Object x = items[takeptr]; if (++takeptr == 100) takeptr = 0; --count; notFull->**signal()** (p. 719); 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. 715), 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. 715) 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

6.142.2 Constructor & Destructor Documentation

6.142.2.1 virtual decaf::util::concurrent::locks::Condition::~Condition ( ) [inline, virtual]

6.142.3 Member Function Documentation

6.142.3.1 virtual void decaf::util::concurrent::locks::Condition::await ( ) [pure virtual]

Causes the current thread to wait until it is signaled or interrupted.

The lock associated with this **Condition** (p. 715) 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. 719) method for this **Condition** (p. 715) and the current thread happens to be chosen as the thread to be awakened; or
  - Some other thread invokes the **signalAll()** (p. 720) method for this **Condition** (p. 715); 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. 715) 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. 715).
InterruptedException	if the current thread is interrupted (and interruption of thread suspension is supported)
IllegalMonitorState-Exception	if the caller is not the lock owner.

6.142.3.2 virtual bool decaf::util::concurrent::locks::Condition::await (long long *time,* const TimeUnit & *unit*) [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. 715).
InterruptedException	if the current thread is interrupted (and interruption of thread suspension is supported)
IllegalMonitorState-Exception	if the caller is not the lock owner.

6.142.3.3 virtual long long decaf::util::concurrent::locks::Condition::awaitNanos (long long *nanosTimeout*) [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. 719) method for this **Condition** (p. 715) and the current thread happens to be chosen as the thread to be awakened; or
  - Some other thread invokes the **signalAll()** (p. 720) method for this **Condition** (p. 715); 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. 715) 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. 715).
InterruptedException	if the current thread is interrupted (and interruption of thread suspension is supported)
IllegalMonitorStateException	if the caller is not the lock owner.

6.142.3.4 virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly ( ) [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. 719) method for this **Condition** (p. 715) and the current thread happens to be chosen as the thread to be awakened; or
  - Some other thread invokes the **signalAll()** (p. 720) method for this **Condition** (p. 715); 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. 715) 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. 715).
IllegalMonitorState-Exception	if the caller is not the lock owner.

6.142.3.5 virtual bool decaf::util::concurrent::locks::Condition::awaitUntil (const Date & *deadline*) [pure virtual]

6.142.3.6 virtual void decaf::util::concurrent::locks::Condition::signal ( ) [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. 715).

6.142.3.7 virtual void decaf::util::concurrent::locks::Condition::signalAll ( ) [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. 715).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Condition.h**

6.143 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**

6.143.1 Constructor & Destructor Documentation

6.143.1.1 decaf::util::concurrent::ConditionHandle::ConditionHandle ( ) [inline]

6.143.1.2 decaf::util::concurrent::ConditionHandle::~ConditionHandle ( ) [inline]

6.143.1.3 decaf::util::concurrent::ConditionHandle::ConditionHandle ( ) [inline]

6.143.1.4 decaf::util::concurrent::ConditionHandle::~ConditionHandle ( ) [inline]

6.143.2 Field Documentation

6.143.2.1 pthread_cond_t decaf::util::concurrent::ConditionHandle::condition

6.143.2.2 CRITICAL_SECTION decaf::util::concurrent::ConditionHandle::criticalSection

6.143.2.3 volatile unsigned int decaf::util::concurrent::ConditionHandle::generation

6.143.2.4 MutexHandle * decaf::util::concurrent::ConditionHandle::mutex

6.143.2.5 volatile unsigned int decaf::util::concurrent::ConditionHandle::numWaiting

6.143.2.6 volatile unsigned int decaf::util::concurrent::ConditionHandle::numWake

6.143.2.7 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**

6.144 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.

6.144.1 Member Function Documentation

6.144.1.1 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.

6.144.1.2 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.

6.144.1.3 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.

6.144.1.4 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.

6.144.1.5 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.

6.144.1.6 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.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**ConditionImpl.h**

6.145 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 ()

6.145.1 Constructor & Destructor Documentation

6.145.1.1 decaf::net::ConnectException::ConnectException ( ) throw () [inline]

Default Constructor.

6.145.1.2 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

6.145.1.3 decaf::net::ConnectException::ConnectException (const ConnectException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.145.1.4 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

6.145.1.5 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.

6.145.1.6 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

6.145.1.7 virtual decaf::net::ConnectException::~ConnectException ( ) throw () [inline, virtual]

6.145.2 Member Function Documentation

6.145.2.1 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. 1916).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ConnectException.h**

6.146 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** () throw ()
- virtual void **close** ()=0

  Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).
- virtual **const ConnectionMetaData** * **getMetaData** () **const** =0

  Gets the metadata for this connection.
- virtual **Session** * **createSession** ()=0

  *Creates an AUTO_ACKNOWLEDGE **Session** (p. 1830).*
- virtual **Session** * **createSession** (**Session::AcknowledgeMode** ackMode)=0

  *Creates a new **Session** (p. 1830) to work for this **Connection** (p. 725) using the specified acknowledgment mode.*
- virtual std::string **getClientID** () **const** =0

  Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.
- virtual void **setClientID** (**const** std::string &clientID)=0

  Sets the client identifier for this connection.
- 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.

6.146.1 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. 759) object.
- It supports an optional **ExceptionListener** (p. 996) 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

6.146.2 Constructor & Destructor Documentation

6.146.2.1 virtual cms::Connection::~Connection ( ) throw () [virtual]

6.146.3 Member Function Documentation

6.146.3.1 virtual void cms::Connection::close ( ) [pure virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

CMSException *(p. 640)*	

Implements **cms::Closeable** (p. 633).

Implemented in **activemq::core::ActiveMQConnection** (p. 152).

6.146.3.2 virtual Session* cms::Connection::createSession ( ) [pure virtual]

Creates an AUTO_ACKNOWLEDGE **Session** (p. 1830).

Exceptions

CMSException *(p. 640)*	

Implemented in **activemq::core::ActiveMQConnection** (p. 152).

6.146.3.3 virtual Session* cms::Connection::createSession (Session::AcknowledgeMode *ackMode*) [pure virtual]

Creates a new **Session** (p. 1830) to work for this **Connection** (p. 725) using the specified acknowledgment mode.

Parameters

ackMode	the Acknowledgment Mode to use.

Exceptions

CMSException *(p. 640)*	

Implemented in **activemq::core::ActiveMQConnection** (p. 152), and **activemq::core::ActiveMQXAConnection** (p. 335).

6.146.3.4 virtual std::string cms::Connection::getClientID ( ) const [pure virtual]

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

Returns

Client Id String for this **Connection** (p. 725).

Exceptions

CMSException *(p. 640)*	if the provider fails to return the client id or an internal error occurs.

Implemented in **activemq::core::ActiveMQConnection** (p. 154).

6.146.3.5 virtual ExceptionListener* cms::Connection::getExceptionListener ( ) const [pure virtual]

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

Implemented in **activemq::core::ActiveMQConnection** (p. 155).

6.146.3.6 virtual const ConnectionMetaData* cms::Connection::getMetaData ( ) const [pure virtual]

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSException *(p. 640)*	if the provider fails to get the connection metadata for this connection.

See also

ConnectionMetaData (p. 759)

Since

2.0

Implemented in **activemq::core::ActiveMQConnection** (p. 155).

6.146.3.7 virtual void cms::Connection::setClientID (const std::string & *clientID*) [pure virtual]

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 741) object and transparently assigned to the **Connection** (p. 725) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1098).

Parameters

clientID	The unique client identifier to assign to the **Connection** (p. 725).

Exceptions

CMSException *(p. 640)*	if the provider fails to set the client id due to some internal error.
InvalidClientIDException	if the id given is somehow invalid or is a duplicate.
IllegalStateException *(p. 1098)*	if the client tries to set the id after a **Connection** (p. 725) method has been called.

Implemented in **activemq::core::ActiveMQConnection** (p. 161).

6.146.3.8 virtual void cms::Connection::setExceptionListener (ExceptionListener * *listener*) [pure virtual]

Sets the registered Exception Listener for this connection.

Parameters

listener	pointer to and **ExceptionListener** (p. 996)

Implemented in **activemq::core::ActiveMQConnection** (p. 162).

The documentation for this class was generated from the following file:

- src/main/cms/**Connection.h**

6.147 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ConnectionControl** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 **const** std::string & **getConnectedBrokers** () **const**
- virtual std::string & **getConnectedBrokers** ()
- virtual void **setConnectedBrokers** (**const** std::string &**connectedBrokers**)
- virtual **const** std::string & **getReconnectTo** () **const**

- virtual std::string & **getReconnectTo** ()
- virtual void **setReconnectTo** (**const** std::string &**reconnectTo**)
- virtual bool **isRebalanceConnection** () **const**
- virtual void **setRebalanceConnection** (bool **rebalanceConnection**)
- virtual bool **isConnectionControl** () **const**
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor)

  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 Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**
- std::string **connectedBrokers**
- std::string **reconnectTo**
- bool **rebalanceConnection**

6.147.1 Constructor & Destructor Documentation

6.147.1.1 activemq::commands::ConnectionControl::ConnectionControl ( )

6.147.1.2 virtual activemq::commands::ConnectionControl::~ConnectionControl ( ) [virtual]

6.147.2 Member Function Documentation

6.147.2.1 virtual ConnectionControl* activemq::commands::ConnectionControl::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.147.2.2 virtual void activemq::commands::ConnectionControl::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. 382).

6.147.2.3 virtual bool activemq::commands::ConnectionControl::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.147.2.4 virtual const std::string& activemq::commands::ConnectionControl::getConnectedBrokers ( ) const [virtual]

6.147.2.5 virtual std::string& activemq::commands::ConnectionControl::getConnectedBrokers ( ) [virtual]

6.147.2.6 virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.147.2.7 virtual const std::string& activemq::commands::ConnectionControl::getReconnectTo ( ) const [virtual]

6.147.2.8 virtual std::string& activemq::commands::ConnectionControl::getReconnectTo ( ) [virtual]

6.147.2.9 virtual bool activemq::commands::ConnectionControl::isClose ( ) const [virtual]

6.147.2.10 virtual bool activemq::commands::ConnectionControl::isConnectionControl ( ) const [inline, virtual]

Returns

an answer of true to the **isConnectionControl()** (p. 730) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 383).

6.147.2.11 virtual bool activemq::commands::ConnectionControl::isExit ( ) const [virtual]

6.147.2.12 virtual bool activemq::commands::ConnectionControl::isFaultTolerant ( ) const [virtual]

6.147.2.13 virtual bool activemq::commands::ConnectionControl::isRebalanceConnection ( ) const [virtual]

6.147.2.14 virtual bool activemq::commands::ConnectionControl::isResume ( ) const [virtual]

6.147.2.15 virtual bool activemq::commands::ConnectionControl::isSuspend ( ) const [virtual]

6.147.2.16 virtual void **activemq::commands::ConnectionControl::setClose** (bool *close*) [virtual]

6.147.2.17 virtual void **activemq::commands::ConnectionControl::setConnectedBrokers** (const std::string & *connectedBrokers*) [virtual]

6.147.2.18 virtual void **activemq::commands::ConnectionControl::setExit** (bool *exit*) [virtual]

6.147.2.19 virtual void **activemq::commands::ConnectionControl::setFaultTolerant** (bool *faultTolerant*) [virtual]

6.147.2.20 virtual void **activemq::commands::ConnectionControl::setRebalanceConnection** (bool *rebalanceConnection*) [virtual]

6.147.2.21 virtual void **activemq::commands::ConnectionControl::setReconnectTo** (const std::string & *reconnectTo*) [virtual]

6.147.2.22 virtual void **activemq::commands::ConnectionControl::setResume** (bool *resume*) [virtual]

6.147.2.23 virtual void **activemq::commands::ConnectionControl::setSuspend** (bool *suspend*) [virtual]

6.147.2.24 virtual std::string **activemq::commands::ConnectionControl::toString** ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.147.2.25 virtual **Pointer**<**Command**> **activemq::commands::ConnectionControl::visit** (**activemq::state::CommandVisitor** * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.147.3 Field Documentation

6.147.3.1 bool **activemq::commands::ConnectionControl::close** [protected]

6.147.3.2 std::string **activemq::commands::ConnectionControl::connectedBrokers** [protected]

6.147.3.3 bool **activemq::commands::ConnectionControl::exit** [protected]

6.147.3.4 bool **activemq::commands::ConnectionControl::faultTolerant** [protected]

6.147.3.5 const unsigned char **activemq::commands::ConnectionControl::ID_CONNECTIONCONTROL** = 18 [static]

6.147.3.6 bool activemq::commands::ConnectionControl::rebalanceConnection [protected]

6.147.3.7 std::string activemq::commands::ConnectionControl::reconnectTo [protected]

6.147.3.8 bool activemq::commands::ConnectionControl::resume [protected]

6.147.3.9 bool activemq::commands::ConnectionControl::suspend [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionControl.h**

6.148 activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 732).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ConnectionControlMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual ~**ConnectionControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.148.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 732).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.148.2 Constructor & Destructor Documentation

6.148.2.1 activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::-ConnectionControlMarshaller ( ) [inline]

6.148.2.2 virtual activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::∼-ConnectionControlMarshaller ( ) [inline, virtual]

6.148.3 Member Function Documentation

6.148.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::-ConnectionControlMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.148.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConnectionControl-Marshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.148.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionControl-Marshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.148.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller-::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.148.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.148.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.148.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConnectionControlMarshaller.h**

6.149 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ConnectionError** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- **Pointer**< **BrokerError** > **exception**
- **Pointer**< **ConnectionId** > **connectionId**

6.149.1 Constructor & Destructor Documentation

6.149.1.1 activemq::commands::ConnectionError::ConnectionError ( )

6.149.1.2 virtual activemq::commands::ConnectionError::~ConnectionError ( ) [virtual]

6.149.2 Member Function Documentation

6.149.2.1 virtual ConnectionError* activemq::commands::ConnectionError::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.149.2.2 virtual void activemq::commands::ConnectionError::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. 382).

6.149.2.3 virtual bool activemq::commands::ConnectionError::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.149.2.4 virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ( ) const [virtual]

6.149.2.5 virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ( ) [virtual]

6.149.2.6 virtual unsigned char activemq::commands::ConnectionError::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.149.2.7 virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException ( ) const [virtual]

6.149.2.8 virtual Pointer<BrokerError>& activemq::commands::ConnectionError::getException ( ) [virtual]

6.149.2.9 virtual void activemq::commands::ConnectionError::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.149.2.10 virtual void activemq::commands::ConnectionError::setException (const Pointer< BrokerError > & *exception*) [virtual]

6.149.2.11 virtual std::string activemq::commands::ConnectionError::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.149.2.12 virtual Pointer<Command> activemq::commands::ConnectionError::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.149.3 Field Documentation

6.149.3.1 Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId [protected]

6.149.3.2 Pointer<BrokerError> activemq::commands::ConnectionError::exception [protected]

6.149.3.3 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**

6.150 activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 738).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ConnectionErrorMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual ~**ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.150.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 738).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.150.2 Constructor & Destructor Documentation

6.150.2.1 activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::ConnectionErrorMarshaller ( ) [inline]

6.150.2.2 virtual activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::~ConnectionErrorMarshaller ( ) [inline, virtual]

6.150.3 Member Function Documentation

6.150.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.150.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.150.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.150.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.150.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.150.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.150.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConnectionErrorMarshaller.h**

6.151 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p. 725) objects returned implement the CMS **Connection** (p. 725) 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** () throw ()
- virtual **Connection** * **createConnection** ()=0

  Creates a connection with the default user identity.
- virtual **cms::Connection** * **createConnection** (**const** std::string &username, **const** std::string &password)=0

  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

  Creates a connection with the specified user identity.

Static Public Member Functions

- static **ConnectionFactory** * **createCMSConnectionFactory** (**const** std::string &brokerURI)

  Static method that is used to create a provider specific connection factory.

6.151.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p. 725) objects returned implement the CMS **Connection** (p. 725) interface and hide the CMS Provider specific implementation details behind that interface.

A Client creates a new **ConnectionFactory** (p. 741) 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

6.151.2 Constructor & Destructor Documentation

6.151.2.1 virtual cms::ConnectionFactory::~ConnectionFactory ( ) throw () [virtual]

6.151.3 Member Function Documentation

6.151.3.1 static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory (const std::string & *brokerURI*) [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. 741) 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. 741) interface, the caller is responsible for deleting this resource.

Exceptions

CMSException *(p. 640)*	if an internal error occurs while creating the **ConnectionFactory** (p. 741).

6.151.3.2 virtual Connection* cms::ConnectionFactory::createConnection ( ) [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. 1964) 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. 640)*	if an internal error occurs while creating the **Connection** (p. 725).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 168).

6.151.3.3 virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & *username*, const std::string & *password*) [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. 1964) 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

CMSException *(p. 640)*	if an internal error occurs while creating the **Connection** (p. 725).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 168).

6.151.3.4 virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & *username*, const std::string & *password*, const std::string & *clientId*) [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. 1964) 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

CMSException *(p. 640)*	if an internal error occurs while creating the **Connection** (p. 725).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 169).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionFactory.h**

6.152 activemq::exceptions::ConnectionFailedException Class Reference

`#include <src/main/activemq/exceptions/ConnectionFailedException.h>`

Inheritance diagram for activemq::exceptions::ConnectionFailedException:

Public Member Functions

- **ConnectionFailedException** ()
- **ConnectionFailedException** (**const exceptions::ActiveMQException** &ex)
- **ConnectionFailedException** (**const ConnectionFailedException** &ex)
- **ConnectionFailedException** (**const** char ∗file, **const** int lineNumber, **const** char ∗msg,...)
- virtual **ConnectionFailedException** ∗ **clone** () **const**

  Clones this exception.
- virtual ∼**ConnectionFailedException** () throw ()

6.152.1 Constructor & Destructor Documentation

6.152.1.1 activemq::exceptions::ConnectionFailedException::ConnectionFailedException ( ) [inline]

6.152.1.2 activemq::exceptions::ConnectionFailedException::ConnectionFailedException (const exceptions::ActiveMQException & *ex*) [inline]

6.152.1.3 activemq::exceptions::ConnectionFailedException::ConnectionFailedException (const ConnectionFailedException & *ex*) [inline]

6.152.1.4 activemq::exceptions::ConnectionFailedException::ConnectionFailedException (const char ∗ *file,* const int *lineNumber,* const char ∗ *msg, ...*) [inline]

6.152.1.5 virtual activemq::exceptions::ConnectionFailedException::∼ConnectionFailedException ( ) throw () [inline, virtual]

6.152.2 Member Function Documentation

6.152.2.1 virtual ConnectionFailedException∗ activemq::exceptions::ConnectionFailedException::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

Copy of this Exception object

Reimplemented from **activemq::exceptions::ActiveMQException** (p. 204).

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**ConnectionFailedException.h**

6.153 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ConnectionId** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *__value__) **const**

  *Compares the **DataStructure** (p. 877) 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**

6.153.1 Member Typedef Documentation

6.153.1.1 typedef decaf::lang::PointerComparator<ConnectionId> activemq::commands::ConnectionId::COMPARATOR

6.153.2 Constructor & Destructor Documentation

6.153.2.1 activemq::commands::ConnectionId::ConnectionId ( )

6.153.2.2 activemq::commands::ConnectionId::ConnectionId (const ConnectionId & *other*)

6.153.2.3 activemq::commands::ConnectionId::ConnectionId (const SessionId * *sessionId*)

6.153.2.4 activemq::commands::ConnectionId::ConnectionId (const ProducerId * *producerId*)

6.153.2.5 activemq::commands::ConnectionId::ConnectionId (const ConsumerId * *consumerId*)

6.153.2.6 virtual activemq::commands::ConnectionId::~ConnectionId ( ) [virtual]

6.153.3 Member Function Documentation

6.153.3.1 virtual ConnectionId* activemq::commands::ConnectionId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.153.3.2 virtual int activemq::commands::ConnectionId::compareTo (const ConnectionId & *value*) const [virtual]

6.153.3.3 virtual void activemq::commands::ConnectionId::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. 879).

6.153.3.4 virtual bool activemq::commands::ConnectionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.153.3.5 virtual bool activemq::commands::ConnectionId::equals (const ConnectionId & *value*) const [virtual]

6.153.3.6 virtual unsigned char activemq::commands::ConnectionId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.153.3.7 virtual const std::string& activemq::commands::ConnectionId::getValue ( ) const [virtual]

6.153.3.8 virtual std::string& activemq::commands::ConnectionId::getValue ( ) [virtual]

6.153.3.9 virtual bool activemq::commands::ConnectionId::operator< (const ConnectionId & *value*) const [virtual]

6.153.3.10 ConnectionId& activemq::commands::ConnectionId::operator= (const ConnectionId & *other*)

6.153.3.11 virtual bool activemq::commands::ConnectionId::operator== (const ConnectionId & *value*) const [virtual]

6.153.3.12 virtual void activemq::commands::ConnectionId::setValue (const std::string & *value*) [virtual]

6.153.3.13 virtual std::string activemq::commands::ConnectionId::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.153.4 Field Documentation

6.153.4.1 const unsigned char activemq::commands::ConnectionId::ID_CONNECTIONID = 120 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.153.4.2 std::string activemq::commands::ConnectionId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionId.h**

6.154 activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 748).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ConnectionIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual ~**ConnectionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.154.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 748).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.154.2 Constructor & Destructor Documentation

6.154.2.1 activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]

6.154.2.2 virtual activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::~ConnectionIdMarshaller ( ) [inline, virtual]

6.154.3 Member Function Documentation

6.154.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.154.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.154.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.154.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.154.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.154.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.154.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConnectionIdMarshaller.h**

6.155 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ConnectionInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () **const**
- 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 **isFaultTolerant** () **const**
- virtual void **setFaultTolerant** (bool **faultTolerant**)
- virtual bool **isFailoverReconnect** () **const**
- virtual void **setFailoverReconnect** (bool **failoverReconnect**)
- virtual bool **isConnectionInfo** () **const**
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor)

  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 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**
- bool **faultTolerant**
- bool **failoverReconnect**

6.155.1 Constructor & Destructor Documentation

6.155.1.1 activemq::commands::ConnectionInfo::ConnectionInfo ( )

6.155.1.2 virtual activemq::commands::ConnectionInfo::~ConnectionInfo ( ) [virtual]

6.155.2 Member Function Documentation

6.155.2.1 virtual ConnectionInfo* activemq::commands::ConnectionInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.155.2.2 virtual void activemq::commands::ConnectionInfo::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. 382).

6.155.2.3 Pointer<RemoveInfo> activemq::commands::ConnectionInfo::createRemoveCommand ( ) const

6.155.2.4 virtual bool activemq::commands::ConnectionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.155.2.5 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath ( ) const [virtual]

6.155.2.6 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath ( ) [virtual]

6.155.2.7 virtual const std::string& activemq::commands::ConnectionInfo::getClientId ( ) const [virtual]

6.155.2.8 virtual std::string& activemq::commands::ConnectionInfo::getClientId ( ) [virtual]

6.155.2.9 virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId ( ) const [virtual]

6.155.2.10 virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId ( ) [virtual]

6.155.2.11 virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.155.2.12 virtual const std::string& activemq::commands::ConnectionInfo::getPassword ( ) const [virtual]

6.155.2.13 virtual std::string& activemq::commands::ConnectionInfo::getPassword ( ) [virtual]

6.155.2.14 virtual const std::string& activemq::commands::ConnectionInfo::getUserName ( ) const [virtual]

6.155.2.15 virtual std::string& activemq::commands::ConnectionInfo::getUserName ( ) [virtual]

6.155.2.16 virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector ( ) const [virtual]

6.155.2.17 virtual bool activemq::commands::ConnectionInfo::isClientMaster ( ) const [virtual]

6.155.2.18 virtual bool activemq::commands::ConnectionInfo::isConnectionInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isConnectionInfo()** (p. 754) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 383).

6.155.2.19 virtual bool activemq::commands::ConnectionInfo::isFailoverReconnect ( ) const [virtual]

6.155.2.20 virtual bool activemq::commands::ConnectionInfo::isFaultTolerant ( ) const [virtual]

6.155.2.21 virtual bool activemq::commands::ConnectionInfo::isManageable ( ) const [virtual]

6.155.2.22 virtual void activemq::commands::ConnectionInfo::setBrokerMasterConnector (bool *brokerMasterConnector*) [virtual]

6.155.2.23 virtual void activemq::commands::ConnectionInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.155.2.24 virtual void activemq::commands::ConnectionInfo::setClientId (const std::string & *clientId*) [virtual]

6.155.2.25 virtual void activemq::commands::ConnectionInfo::setClientMaster (bool *clientMaster*) [virtual]

6.155.2.26 virtual void activemq::commands::ConnectionInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.155.2.27 virtual void activemq::commands::ConnectionInfo::setFailoverReconnect (bool *failoverReconnect*) [virtual]

6.155.2.28 virtual void activemq::commands::ConnectionInfo::setFaultTolerant (bool *faultTolerant*) [virtual]

6.155.2.29 virtual void activemq::commands::ConnectionInfo::setManageable (bool *manageable*) [virtual]

6.155.2.30 virtual void activemq::commands::ConnectionInfo::setPassword (const std::string & *password*) [virtual]

6.155.2.31 virtual void activemq::commands::ConnectionInfo::setUserName (const std::string & *userName*) [virtual]

6.155.2.32 virtual std::string activemq::commands::ConnectionInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.155.2.33 virtual Pointer<Command> activemq::commands::ConnectionInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.155.3 Field Documentation

6.155.3.1 bool activemq::commands::ConnectionInfo::brokerMasterConnector [protected]

6.155.3.2 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConnectionInfo::brokerPath [protected]

6.155.3.3 std::string activemq::commands::ConnectionInfo::clientId [protected]

6.155.3.4 bool activemq::commands::ConnectionInfo::clientMaster [protected]

6.155.3.5 Pointer<ConnectionId> activemq::commands::ConnectionInfo::connectionId [protected]

6.155.3.6 bool activemq::commands::ConnectionInfo::failoverReconnect [protected]

6.155.3.7 bool activemq::commands::ConnectionInfo::faultTolerant [protected]

6.155.3.8 const unsigned char activemq::commands::ConnectionInfo::ID_CONNECTIONINFO = 3 [static]

6.155.3.9 bool activemq::commands::ConnectionInfo::manageable [protected]

6.155.3.10 std::string activemq::commands::ConnectionInfo::password [protected]

6.155.3.11 std::string activemq::commands::ConnectionInfo::userName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionInfo.h**

6.156 activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 756).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ConnectionInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual ∼**ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.156.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 756).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.156.2 Constructor & Destructor Documentation

6.156.2.1 activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::ConnectionInfoMarshaller ( ) [inline]

6.156.2.2 virtual activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::∼ConnectionInfoMarshaller ( ) [inline, virtual]

6.156.3 Member Function Documentation

6.156.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.156.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.156.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.156.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.156.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.156.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.156.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConnectionInfoMarshaller.h**

6.157 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p. 759) object provides information describing the **Connection** (p. 725) object.

`#include <src/main/cms/ConnectionMetaData.h>`

Inheritance diagram for cms::ConnectionMetaData:

Public Member Functions

- virtual ~**ConnectionMetaData** () throw ()
- virtual std::string **getCMSVersion** () **const** =0

  Gets the CMS API version.
- virtual int **getCMSMajorVersion** () **const** =0

  Gets the CMS major version number.
- virtual int **getCMSMinorVersion** () **const** =0

  Gets the CMS minor version number.
- virtual std::string **getCMSProviderName** () **const** =0

  Gets the CMS provider name.
- virtual std::string **getProviderVersion** () **const** =0

  Gets the CMS provider version.
- virtual int **getProviderMajorVersion** () **const** =0

  Gets the CMS provider major version number.
- virtual int **getProviderMinorVersion** () **const** =0

  Gets the CMS provider minor version number.
- virtual std::vector< std::string > **getCMSXPropertyNames** () **const** =0

  Gets an Vector of the CMSX property names.

6.157.1 Detailed Description

A **ConnectionMetaData** (p. 759) object provides information describing the **Connection** (p. 725) object.

Since

1.3

6.157.2 Constructor & Destructor Documentation

6.157.2.1 virtual cms::ConnectionMetaData::~ConnectionMetaData ( ) throw () [virtual]

6.157.3 Member Function Documentation

6.157.3.1 virtual int cms::ConnectionMetaData::getCMSMajorVersion ( ) const [pure virtual]

Gets the CMS major version number.

Returns

the CMS API major version number

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 176).

6.157.3.2 virtual int cms::ConnectionMetaData::getCMSMinorVersion ( ) const [pure virtual]

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 176).

6.157.3.3 virtual std::string cms::ConnectionMetaData::getCMSProviderName ( ) const [pure virtual]

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 177).

6.157.3.4 virtual std::string cms::ConnectionMetaData::getCMSVersion ( ) const [pure virtual]

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 177).

6.157.3.5 virtual std::vector<std::string> cms::ConnectionMetaData::getCMSXPropertyNames ( ) const [pure virtual]

Gets an Vector of the CMSX property names.

Returns

an Vector of CMSX property names

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 177).

6.157.3.6 virtual int cms::ConnectionMetaData::getProviderMajorVersion ( ) const [pure virtual]

Gets the CMS provider major version number.

Returns

the CMS provider major version number

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 177).

6.157.3.7 virtual int cms::ConnectionMetaData::getProviderMinorVersion ( ) const [pure virtual]

Gets the CMS provider minor version number.

Returns

the CMS provider minor version number

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 178).

6.157.3.8 virtual std::string cms::ConnectionMetaData::getProviderVersion ( ) const [pure virtual]

Gets the CMS provider version.

Returns

the CMS provider version

Exceptions

CMSException *(p. 640)*	If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 178).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionMetaData.h**

6.158 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 LinkedList**< **Pointer** < **DestinationInfo** > > & **getTempDesinations** () **const**
- std::vector< **Pointer** < **SessionState** > > **getSessionStates** () **const**
- **StlMap**< **Pointer**< **ConsumerId** > , **Pointer**< **ConsumerInfo** > , **ConsumerId::COMPARATOR** > & **getRecoveringPullConsumers** ()
- void **setConnectionInterruptProcessingComplete** (bool connectionInterruptProcessingComplete)
- bool **isConnectionInterruptProcessingComplete** ()

6.158.1 Constructor & Destructor Documentation

6.158.1.1 activemq::state::ConnectionState::ConnectionState (const Pointer< ConnectionInfo > & *info*)

6.158.1.2 virtual activemq::state::ConnectionState::~ConnectionState ( ) [virtual]

6.158.2 Member Function Documentation

6.158.2.1 void activemq::state::ConnectionState::addSession (const Pointer< SessionInfo > & *info*) [inline]

References activemq::commands::SessionInfo::getSessionId().

6.158.2.2 void activemq::state::ConnectionState::addTempDestination (const Pointer< DestinationInfo > & *info*) [inline]

6.158.2.3 void activemq::state::ConnectionState::addTransactionState (const Pointer< TransactionId > & *id*) [inline]

6.158.2.4 void activemq::state::ConnectionState::checkShutdown ( ) const

6.158.2.5 const Pointer<commands::ConnectionInfo>& activemq::state::ConnectionState::getInfo ( ) const [inline]

6.158.2.6 StlMap< Pointer<ConsumerId>, Pointer<ConsumerInfo>, ConsumerId::COMPARATOR >& activemq::state::ConnectionState::getRecoveringPullConsumers ( ) [inline]

6.158.2.7 const Pointer<SessionState>& activemq::state::ConnectionState::getSessionState (const Pointer< SessionId > & *id*) const [inline]

6.158.2.8 std::vector< Pointer<SessionState> > activemq::state::ConnectionState::getSessionStates ( ) const [inline]

6.158.2.9 const LinkedList< Pointer<DestinationInfo> >& activemq::state::ConnectionState::getTempDesinations ( ) const [inline]

6.158.2.10 const Pointer<TransactionState>& activemq::state::ConnectionState::getTransactionState (const Pointer< TransactionId > & *id*) const [inline]

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

6.158.2.11 std::vector< Pointer<TransactionState> > activemq::state::ConnectionState::getTransactionStates ( ) const [inline]

6.158.2.12 bool activemq::state::ConnectionState::isConnectionInterruptProcessingComplete ( ) [inline]

6.158.2.13 Pointer<SessionState> activemq::state::ConnectionState::removeSession (const Pointer< SessionId > & *id*) [inline]

6.158.2.14 void activemq::state::ConnectionState::removeTempDestination (const Pointer< ActiveMQDestination > & *destination*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get(), and activemq::commands::DestinationInfo::getDestination().

6.158.2.15 Pointer<TransactionState> activemq::state::ConnectionState::removeTransactionState (const Pointer< TransactionId > & *id*) [inline]

6.158.2.16 void activemq::state::ConnectionState::reset (const Pointer< ConnectionInfo > & *info*)

6.158.2.17 void activemq::state::ConnectionState::setConnectionInterruptProcessingComplete (bool *connectionInterruptProcessingComplete*) [inline]

6.158.2.18 void activemq::state::ConnectionState::shutdown ( )

6.158.2.19 std::string activemq::state::ConnectionState::toString ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConnectionState.h**

6.159 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)
- void **trackBack** (**const Pointer**< **Command** > &command)
- void **restore** (**const Pointer**< **transport::Transport** > &transport)
- void **connectionInterruptProcessingComplete** (**transport::Transport** *transport, **const Pointer**< **ConnectionId** > &connectionId)
- void **transportInterrupted** ()
- virtual **Pointer**< **Command** > **processDestinationInfo** (**DestinationInfo** *info)
- virtual **Pointer**< **Command** > **processRemoveDestination** (**DestinationInfo** *info)
- virtual **Pointer**< **Command** > **processProducerInfo** (**ProducerInfo** *info)
- virtual **Pointer**< **Command** > **processRemoveProducer** (**ProducerId** *id)
- virtual **Pointer**< **Command** > **processConsumerInfo** (**ConsumerInfo** *info)
- virtual **Pointer**< **Command** > **processRemoveConsumer** (**ConsumerId** *id)
- virtual **Pointer**< **Command** > **processSessionInfo** (**SessionInfo** *info)
- virtual **Pointer**< **Command** > **processRemoveSession** (**SessionId** *id)
- virtual **Pointer**< **Command** > **processConnectionInfo** (**ConnectionInfo** *info)
- virtual **Pointer**< **Command** > **processRemoveConnection** (**ConnectionId** *id)
- virtual **Pointer**< **Command** > **processMessage** (**Message** *message)
- virtual **Pointer**< **Command** > **processMessageAck** (**MessageAck** *ack)
- virtual **Pointer**< **Command** > **processBeginTransaction** (**TransactionInfo** *info)
- virtual **Pointer**< **Command** > **processPrepareTransaction** (**TransactionInfo** *info)
- virtual **Pointer**< **Command** > **processCommitTransactionOnePhase** (**TransactionInfo** *info)
- virtual **Pointer**< **Command** > **processCommitTransactionTwoPhase** (**TransactionInfo** *info)
- virtual **Pointer**< **Command** > **processRollbackTransaction** (**TransactionInfo** *info)
- virtual **Pointer**< **Command** > **processEndTransaction** (**TransactionInfo** *info)
- 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)
- bool **isTrackTransactionProducers** () **const**
- void **setTrackTransactionProducers** (bool trackTransactionProducers)

Friends

- class **RemoveTransactionAction**

6.159.1 Constructor & Destructor Documentation

6.159.1.1 activemq::state::ConnectionStateTracker::ConnectionStateTracker ( )

6.159.1.2 virtual activemq::state::ConnectionStateTracker::~ConnectionStateTracker ( ) [virtual]

6.159.2 Member Function Documentation

6.159.2.1 void activemq::state::ConnectionStateTracker::connectionInterruptProcessingComplete (transport::Transport * *transport,* const Pointer< ConnectionId > & *connectionId*)

6.159.2.2 int activemq::state::ConnectionStateTracker::getMaxCacheSize ( ) const [inline]

6.159.2.3 bool activemq::state::ConnectionStateTracker::isRestoreConsumers ( ) const [inline]

6.159.2.4 bool activemq::state::ConnectionStateTracker::isRestoreProducers ( ) const [inline]

6.159.2.5 bool activemq::state::ConnectionStateTracker::isRestoreSessions ( ) const [inline]

6.159.2.6 bool activemq::state::ConnectionStateTracker::isRestoreTransaction ( ) const [inline]

6.159.2.7 bool activemq::state::ConnectionStateTracker::isTrackMessages ( ) const [inline]

6.159.2.8 bool activemq::state::ConnectionStateTracker::isTrackTransactionProducers ( ) const [inline]

6.159.2.9 bool activemq::state::ConnectionStateTracker::isTrackTransactions ( ) const [inline]

6.159.2.10 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processBeginTransaction (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 677).

6.159.2.11 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionOnePhase (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 677).

6.159.2.12 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 677).

6.159.2.13 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo (ConnectionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 678).

6.159.2.14 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo (ConsumerInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 678).

6.159.2.15 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo (DestinationInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 678).

6.159.2.16 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 678).

6.159.2.17 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage (Message * *message*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 678).

6.159.2.18 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck (MessageAck * *ack*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 679).

6.159.2.19 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepare-Transaction (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 679).

6.159.2.20 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo (ProducerInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 679).

6.159.2.21 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemove-Connection (ConnectionId * *id*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 679).

6.159.2.22 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer (ConsumerId * *id*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 679).

6.159.2.23 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemove-Destination (DestinationInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 680).

6.159.2.24 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer (ProducerId * *id*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 680).

6.159.2.25 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession (SessionId * *id*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 680).

6.159.2.26 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollback-Transaction (TransactionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 680).

6.159.2.27 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo (SessionInfo * *info*) [virtual]

Implements **activemq::state::CommandVisitor** (p. 680).

6.159.2.28 void activemq::state::ConnectionStateTracker::restore (const Pointer< transport::Transport > & *transport*)

6.159.2.29 void activemq::state::ConnectionStateTracker::setMaxCacheSize (int *maxCacheSize*) [inline]

6.159.2.30 void activemq::state::ConnectionStateTracker::setRestoreConsumers (bool *restoreConsumers*) [inline]

6.159.2.31 void activemq::state::ConnectionStateTracker::setRestoreProducers (bool *restoreProducers*) [inline]

6.159.2.32 void activemq::state::ConnectionStateTracker::setRestoreSessions (bool *restoreSessions*) [inline]

6.159.2.33 void activemq::state::ConnectionStateTracker::setRestoreTransaction (bool *restoreTransaction*) [inline]

6.159.2.34 void activemq::state::ConnectionStateTracker::setTrackMessages (bool *trackMessages*) [inline]

6.159.2.35 void activemq::state::ConnectionStateTracker::setTrackTransactionProducers (bool *trackTransactionProducers*) [inline]

6.159.2.36 void activemq::state::ConnectionStateTracker::setTrackTransactions (bool *trackTransactions*) [inline]

6.159.2.37 Pointer<Tracked> activemq::state::ConnectionStateTracker::track (const Pointer< Command > & *command*)

6.159.2.38 void activemq::state::ConnectionStateTracker::trackBack (const Pointer< Command > & *command*)

6.159.2.39 void activemq::state::ConnectionStateTracker::transportInterrupted ( )

6.159.3 Friends And Related Function Documentation

6.159.3.1 friend class RemoveTransactionAction [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConnectionStateTracker.h**

6.160 decaf::util::logging::ConsoleHandler Class Reference

This **Handler** (p. 1085) publishes log records to System.err.

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

Inheritance diagram for decaf::util::logging::ConsoleHandler:

Public Member Functions

- **ConsoleHandler** ()
- virtual ~**ConsoleHandler** ()
- virtual void **close** ()

  Close the current output stream.
- virtual void **publish** (**const LogRecord** &record)

  *Publish the Log Record to this **Handler** (p. 1085).*

6.160.1 Detailed Description

This **Handler** (p. 1085) publishes log records to System.err.

By default the **SimpleFormatter** (p. 1891) is used to generate brief summaries.

Configuration: By default each **ConsoleHandler** (p. 768) is initialized using the following **LogManager** (p. 1327) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

ConsoleHandler.level specifies the default level for the **Handler** (p. 1085) (defaults to **Level.INFO** (p. 1256)). ConsoleHandler.filter specifies the name of a **Filter** (p. 1031) class to use (defaults to no **Filter** (p. 1031)). ConsoleHandler.formatter specifies the name of a **Formatter** (p. 1074) class to use (defaults to **SimpleFormatter** (p. 1891)).

Since

1.0

6.160.2 Constructor & Destructor Documentation

6.160.2.1 decaf::util::logging::ConsoleHandler::ConsoleHandler ( )

6.160.2.2 virtual decaf::util::logging::ConsoleHandler::~ConsoleHandler ( ) [inline, virtual]

6.160.3 Member Function Documentation

6.160.3.1 virtual void decaf::util::logging::ConsoleHandler::close ( ) [virtual]

Close the current output stream.

Override the **StreamHandler** (p. 2017) close to flush the Std Err stream but doesn't close.

Exceptions

IOException	

Reimplemented from **decaf::util::logging::StreamHandler** (p. 2019).

6.160.3.2 virtual void decaf::util::logging::ConsoleHandler::publish (const LogRecord & *record*) [virtual]

Publish the Log Record to this **Handler** (p. 1085).

Parameters

record	The **LogRecord** (p. 1333) to Publish

Reimplemented from **decaf::util::logging::StreamHandler** (p. 2019).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**ConsoleHandler.h**

6.161 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ConsumerControl** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 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)

  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 Attributes

- **Pointer< ActiveMQDestination >** **destination**
- bool **close**
- **Pointer< ConsumerId >** **consumerId**
- int **prefetch**
- bool **flush**
- bool **start**
- bool **stop**

6.161.1 Constructor & Destructor Documentation

6.161.1.1 activemq::commands::ConsumerControl::ConsumerControl ( )

6.161.1.2 virtual activemq::commands::ConsumerControl::~ConsumerControl ( ) [virtual]

6.161.2 Member Function Documentation

6.161.2.1 virtual ConsumerControl* activemq::commands::ConsumerControl::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.161.2.2 virtual void activemq::commands::ConsumerControl::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. 382).

6.161.2.3 virtual bool activemq::commands::ConsumerControl::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.161.2.4 virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ( ) const [virtual]

6.161.2.5 virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ( ) [virtual]

6.161.2.6 virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.161.2.7 virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination ( ) const [virtual]

6.161.2.8 virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination ( ) [virtual]

6.161.2.9 virtual int activemq::commands::ConsumerControl::getPrefetch ( ) const [virtual]

6.161.2.10 virtual bool activemq::commands::ConsumerControl::isClose ( ) const [virtual]

6.161.2.11 virtual bool activemq::commands::ConsumerControl::isFlush ( ) const [virtual]

6.161.2.12 virtual bool activemq::commands::ConsumerControl::isStart ( ) const [virtual]

6.161.2.13 virtual bool activemq::commands::ConsumerControl::isStop ( ) const [virtual]

6.161.2.14 virtual void activemq::commands::ConsumerControl::setClose (bool *close*) [virtual]

6.161.2.15 virtual void activemq::commands::ConsumerControl::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.161.2.16 virtual void activemq::commands::ConsumerControl::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.161.2.17 virtual void activemq::commands::ConsumerControl::setFlush (bool *flush*) [virtual]

6.161.2.18 virtual void activemq::commands::ConsumerControl::setPrefetch (int *prefetch*) [virtual]

6.161.2.19 virtual void activemq::commands::ConsumerControl::setStart (bool *start*) [virtual]

6.161.2.20 virtual void activemq::commands::ConsumerControl::setStop (bool *stop*) [virtual]

6.161.2.21 virtual std::string activemq::commands::ConsumerControl::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.161.2.22 virtual Pointer<Command> activemq::commands::ConsumerControl::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.161.3 Field Documentation

6.161.3.1 bool activemq::commands::ConsumerControl::close [protected]

6.161.3.2 Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId [protected]

6.161.3.3 Pointer<ActiveMQDestination> activemq::commands::ConsumerControl::destination [protected]

6.161.3.4 bool activemq::commands::ConsumerControl::flush [protected]

6.161.3.5 const unsigned char activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17 [static]

6.161.3.6 int activemq::commands::ConsumerControl::prefetch [protected]

6.161.3.7 bool activemq::commands::ConsumerControl::start [protected]

6.161.3.8 bool activemq::commands::ConsumerControl::stop [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConsumerControl.h**

6.162 activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 773).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ConsumerControlMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual ~**ConsumerControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.162.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 773).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.162.2 Constructor & Destructor Documentation

6.162.2.1 activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::ConsumerControlMarshaller ( ) [inline]

6.162.2.2 virtual activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::~ConsumerControlMarshaller ( ) [inline, virtual]

6.162.3 Member Function Documentation

6.162.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.162.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.162.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.162.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.162.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.162.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.162.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConsumerControlMarshaller.h**

6.163 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 consumerIdd)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ConsumerId** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *__value__) **const**

  *Compares the **DataStructure** (p. 877) 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**

6.163.1 Member Typedef Documentation

6.163.1.1 typedef decaf::lang::PointerComparator<ConsumerId> activemq::commands::ConsumerId::COMPARATOR

6.163.2 Constructor & Destructor Documentation

6.163.2.1 activemq::commands::ConsumerId::ConsumerId ( )

6.163.2.2 activemq::commands::ConsumerId::ConsumerId (const ConsumerId & *other*)

6.163.2.3 activemq::commands::ConsumerId::ConsumerId (const SessionId & *sessionId,* long long *consumerIdd*)

6.163.2.4 virtual activemq::commands::ConsumerId::~ConsumerId ( ) [virtual]

6.163.3 Member Function Documentation

6.163.3.1 virtual ConsumerId* activemq::commands::ConsumerId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.163.3.2 virtual int activemq::commands::ConsumerId::compareTo (const ConsumerId & *value*) const [virtual]

6.163.3.3 virtual void activemq::commands::ConsumerId::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. 879).

6.163.3.4 virtual bool activemq::commands::ConsumerId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.163.3.5 virtual bool activemq::commands::ConsumerId::equals (const ConsumerId & *value*) const [virtual]

6.163.3.6 virtual const std::string& activemq::commands::ConsumerId::getConnectionId ( ) const [virtual]

6.163.3.7 virtual std::string& activemq::commands::ConsumerId::getConnectionId ( ) [virtual]

6.163.3.8 virtual unsigned char activemq::commands::ConsumerId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.163.3.9 const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId ( ) const

6.163.3.10 virtual long long activemq::commands::ConsumerId::getSessionId ( ) const [virtual]

6.163.3.11 virtual long long activemq::commands::ConsumerId::getValue ( ) const [virtual]

6.163.3.12 virtual bool activemq::commands::ConsumerId::operator< (const ConsumerId & *value*) const [virtual]

6.163.3.13 ConsumerId& activemq::commands::ConsumerId::operator= (const ConsumerId & *other*)

6.163.3.14 virtual bool activemq::commands::ConsumerId::operator== (const ConsumerId & *value*) const [virtual]

6.163.3.15 virtual void activemq::commands::ConsumerId::setConnectionId (const std::string & *connectionId*) [virtual]

6.163.3.16 virtual void activemq::commands::ConsumerId::setSessionId (long long *sessionId*) [virtual]

6.163.3.17 virtual void activemq::commands::ConsumerId::setValue (long long *value*) [virtual]

6.163.3.18 virtual std::string activemq::commands::ConsumerId::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.163.4 Field Documentation

6.163.4.1 std::string activemq::commands::ConsumerId::connectionId [protected]

6.163.4.2 const unsigned char activemq::commands::ConsumerId::ID_CONSUMERID = 122 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.163.4.3 long long activemq::commands::ConsumerId::sessionId [protected]

6.163.4.4 long long activemq::commands::ConsumerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConsumerId.h**

6.164 activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 779).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ConsumerIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual ∼**ConsumerIdMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.164.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 779).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.164.2 Constructor & Destructor Documentation

6.164.2.1 activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::ConsumerIdMarshaller ( ) [inline]

6.164.2.2 virtual activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::~ConsumerIdMarshaller ( ) [inline, virtual]

6.164.3 Member Function Documentation

6.164.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.164.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.164.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.164.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.164.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.164.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.164.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConsumerIdMarshaller.h**

6.165 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ConsumerInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () **const**
- 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)

  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 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**

6.165.1 Constructor & Destructor Documentation

6.165.1.1 activemq::commands::ConsumerInfo::ConsumerInfo ( )

6.165.1.2 virtual activemq::commands::ConsumerInfo::~ConsumerInfo ( ) [virtual]

6.165.2 Member Function Documentation

6.165.2.1 virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.165.2.2 virtual void activemq::commands::ConsumerInfo::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. 382).

6.165.2.3 Pointer<RemoveInfo> activemq::commands::ConsumerInfo::createRemoveCommand ( ) const

6.165.2.4 virtual bool activemq::commands::ConsumerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.165.2.5 virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate ( ) const [virtual]

6.165.2.6 virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate ( ) [virtual]

6.165.2.7 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath ( ) const [virtual]

6.165.2.8 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath ( ) [virtual]

6.165.2.9 virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId ( ) const [virtual]

Referenced by activemq::state::SessionState::addConsumer().

6.165.2.10 virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId ( ) [virtual]

6.165.2.11 virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.165.2.12 virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination ( ) const [virtual]

6.165.2.13 virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination ( ) [virtual]

6.165.2.14 virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit ( ) const [virtual]

6.165.2.15 virtual const std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath ( ) const [virtual]

6.165.2.16 virtual std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath ( ) [virtual]

6.165.2.17 virtual int activemq::commands::ConsumerInfo::getPrefetchSize ( ) const [virtual]

6.165.2.18 virtual unsigned char activemq::commands::ConsumerInfo::getPriority ( ) const [virtual]

6.165.2.19 virtual const std::string& activemq::commands::ConsumerInfo::getSelector ( ) const [virtual]

6.165.2.20 virtual std::string& activemq::commands::ConsumerInfo::getSelector ( ) [virtual]

6.165.2.21 virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName ( ) const [virtual]

6.165.2.22 virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName ( ) [virtual]

6.165.2.23 virtual bool activemq::commands::ConsumerInfo::isBrowser ( ) const [virtual]

6.165.2.24 virtual bool activemq::commands::ConsumerInfo::isConsumerInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isConsumerInfo()** (p. 786) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 383).

6.165.2.25 virtual bool activemq::commands::ConsumerInfo::isDispatchAsync ( ) const [virtual]

6.165.2.26 virtual bool activemq::commands::ConsumerInfo::isExclusive ( ) const [virtual]

6.165.2.27 virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription ( ) const [virtual]

6.165.2.28 virtual bool activemq::commands::ConsumerInfo::isNoLocal ( ) const [virtual]

6.165.2.29 virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks ( ) const [virtual]

6.165.2.30 virtual bool activemq::commands::ConsumerInfo::isOptimizedAcknowledge ( ) const [virtual]

6.165.2.31 virtual bool activemq::commands::ConsumerInfo::isRetroactive ( ) const [virtual]

6.165.2.32 virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate (const Pointer< BooleanExpression > & *additionalPredicate*) [virtual]

6.165.2.33 virtual void **activemq::commands::ConsumerInfo::setBrokerPath** (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.165.2.34 virtual void **activemq::commands::ConsumerInfo::setBrowser** (bool *browser*) [virtual]

6.165.2.35 virtual void **activemq::commands::ConsumerInfo::setConsumerId** (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.165.2.36 virtual void **activemq::commands::ConsumerInfo::setDestination** (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.165.2.37 virtual void **activemq::commands::ConsumerInfo::setDispatchAsync** (bool *dispatchAsync*) [virtual]

6.165.2.38 virtual void **activemq::commands::ConsumerInfo::setExclusive** (bool *exclusive*) [virtual]

6.165.2.39 virtual void **activemq::commands::ConsumerInfo::setMaximumPendingMessageLimit** (int *maximumPendingMessageLimit*) [virtual]

6.165.2.40 virtual void **activemq::commands::ConsumerInfo::setNetworkConsumerPath** (const std::vector< decaf::lang::Pointer< ConsumerId > > & *networkConsumerPath*) [virtual]

6.165.2.41 virtual void **activemq::commands::ConsumerInfo::setNetworkSubscription** (bool *networkSubscription*) [virtual]

6.165.2.42 virtual void **activemq::commands::ConsumerInfo::setNoLocal** (bool *noLocal*) [virtual]

6.165.2.43 virtual void **activemq::commands::ConsumerInfo::setNoRangeAcks** (bool *noRangeAcks*) [virtual]

6.165.2.44 virtual void **activemq::commands::ConsumerInfo::setOptimizedAcknowledge** (bool *optimizedAcknowledge*) [virtual]

6.165.2.45 virtual void **activemq::commands::ConsumerInfo::setPrefetchSize** (int *prefetchSize*) [virtual]

6.165.2.46 virtual void **activemq::commands::ConsumerInfo::setPriority** (unsigned char *priority*) [virtual]

6.165.2.47 virtual void **activemq::commands::ConsumerInfo::setRetroactive** (bool *retroactive*) [virtual]

6.165.2.48 virtual void **activemq::commands::ConsumerInfo::setSelector** (const std::string & *selector*) [virtual]

6.165.2.49 virtual void **activemq::commands::ConsumerInfo::setSubscriptionName** (const std::string & *subscriptionName*) [virtual]

6.165.2.50 virtual std::string **activemq::commands::ConsumerInfo::toString** ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.165.2.51 virtual **Pointer**<**Command**> **activemq::commands::ConsumerInfo::visit (** **activemq::state::CommandVisitor** * ***visitor*** **)** [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.165.3 Field Documentation

6.165.3.1 **Pointer**<**BooleanExpression**> **activemq::commands::ConsumerInfo::additionalPredicate** [protected]

6.165.3.2 std::vector< **decaf::lang::Pointer**<**BrokerId**> > **activemq::commands::ConsumerInfo::brokerPath** [protected]

6.165.3.3 bool **activemq::commands::ConsumerInfo::browser** [protected]

6.165.3.4 **Pointer**<**ConsumerId**> **activemq::commands::ConsumerInfo::consumerId** [protected]

6.165.3.5 **Pointer**<**ActiveMQDestination**> **activemq::commands::ConsumerInfo::destination** [protected]

6.165.3.6 bool **activemq::commands::ConsumerInfo::dispatchAsync** [protected]

6.165.3.7 bool **activemq::commands::ConsumerInfo::exclusive** [protected]

6.165.3.8 **const** unsigned char **activemq::commands::ConsumerInfo::ID_CONSUMERINFO = 5** [static]

6.165.3.9 int **activemq::commands::ConsumerInfo::maximumPendingMessageLimit** [protected]

6.165.3.10 std::vector< **decaf::lang::Pointer**<**ConsumerId**> > **activemq::commands::ConsumerInfo-::networkConsumerPath** [protected]

6.165.3.11 bool **activemq::commands::ConsumerInfo::networkSubscription** [protected]

6.165.3.12 bool **activemq::commands::ConsumerInfo::noLocal** [protected]

6.165.3.13 bool **activemq::commands::ConsumerInfo::noRangeAcks** [protected]

6.165.3.14 bool **activemq::commands::ConsumerInfo::optimizedAcknowledge** [protected]

6.165.3.15 int **activemq::commands::ConsumerInfo::prefetchSize** [protected]

6.165.3.16 unsigned char **activemq::commands::ConsumerInfo::priority** [protected]

6.165.3.17 bool **activemq::commands::ConsumerInfo::retroactive** [protected]

6.165.3.18 std::string **activemq::commands::ConsumerInfo::selector** [protected]

6.165.3.19 std::string activemq::commands::ConsumerInfo::subscriptionName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConsumerInfo.h**

6.166 activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 789).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ConsumerInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual ~**ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.166.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 789).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.166.2 Constructor & Destructor Documentation

6.166.2.1 activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]

6.166.2.2 virtual activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::~ConsumerInfoMarshaller ( ) [inline, virtual]

6.166.3 Member Function Documentation

6.166.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.166.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.166.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.166.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.166.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.166.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.166.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ConsumerInfoMarshaller.h**

6.167 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**

6.167.1 Constructor & Destructor Documentation

6.167.1.1 activemq::state::ConsumerState::ConsumerState (const Pointer< ConsumerInfo > & *info*)

6.167.1.2 virtual activemq::state::ConsumerState::~ConsumerState ( ) [virtual]

6.167.2 Member Function Documentation

6.167.2.1 const Pointer<ConsumerInfo>& activemq::state::ConsumerState::getInfo ( ) const [inline]

6.167.2.2 std::string activemq::state::ConsumerState::toString ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConsumerState.h**

6.168 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ControlCommand** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- std::string **command**

6.168.1 Constructor & Destructor Documentation

6.168.1.1 activemq::commands::ControlCommand::ControlCommand ( )

6.168.1.2 virtual activemq::commands::ControlCommand::~ControlCommand ( ) [virtual]

6.168.2 Member Function Documentation

6.168.2.1 virtual ControlCommand* activemq::commands::ControlCommand::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.168.2.2 virtual void activemq::commands::ControlCommand::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. 382).

6.168.2.3 virtual bool activemq::commands::ControlCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.168.2.4 virtual const std::string& activemq::commands::ControlCommand::getCommand ( ) const [virtual]

6.168.2.5 virtual std::string& activemq::commands::ControlCommand::getCommand ( ) [virtual]

6.168.2.6 virtual unsigned char activemq::commands::ControlCommand::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.168.2.7 virtual void activemq::commands::ControlCommand::setCommand (const std::string & *command*) [virtual]

6.168.2.8 virtual std::string activemq::commands::ControlCommand::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.168.2.9 virtual Pointer<Command> activemq::commands::ControlCommand::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.168.3 Field Documentation

6.168.3.1 std::string activemq::commands::ControlCommand::command [protected]

6.168.3.2 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**

6.169 activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 795).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ControlCommandMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual ~**ControlCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.169.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 795).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.169.2 Constructor & Destructor Documentation

6.169.2.1 activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::ControlCommandMarshaller ( ) [inline]

6.169.2.2 virtual activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::∼ControlCommandMarshaller ( ) [inline, virtual]

6.169.3 Member Function Documentation

6.169.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.169.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.169.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::looseMarshal (OpenWireFormat ∗ *format,* commands::DataStructure ∗ *command,* decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.169.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.169.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.169.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.169.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ControlCommandMarshaller.h**

6.170 decaf::util::concurrent::CopyOnWriteArrayList< E > Class Template Reference

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

Inheritance diagram for decaf::util::concurrent::CopyOnWriteArrayList< E >:

Data Structures

- class **ArrayListIterator**

Public Member Functions

- **CopyOnWriteArrayList** ()
- **CopyOnWriteArrayList** (**const Collection**< E > &collection)
- **CopyOnWriteArrayList** (**const CopyOnWriteArrayList**< E > &collection)
- **CopyOnWriteArrayList** (**const** E *array, int **size**)
- virtual ~**CopyOnWriteArrayList** ()
- **CopyOnWriteArrayList**< E > & **operator=** (**const CopyOnWriteArrayList**< E > &list)
- **CopyOnWriteArrayList**< E > & **operator=** (**const Collection**< E > &list)
- virtual void **copy** (**const Collection**< E > &collection)

  *Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).*

- virtual bool **add** (**const** E &value)

  Returns true if this collection changed as a result of the call.
- virtual bool **addAll** (**const Collection**< E > &collection)

  Adds all of the elements in the specified collection to this collection.
- virtual void **clear** ()

  Removes all of the elements from this collection (optional operation).
- virtual bool **contains** (**const** E &value) **const**

  Returns true if this collection contains the specified element.
- virtual bool **containsAll** (**const Collection**< E > &collection) **const**

  Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **equals** (**const Collection**< E > &collection) **const**

  Compares the passed collection to this one, if they contain the same elements, i.e.
- virtual bool **isEmpty** () **const**
- virtual bool **remove** (**const** E &value)

  Removes a single instance of the specified element from the collection.
- virtual bool **removeAll** (**const Collection**< E > &collection)

  Removes all this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (**const Collection**< E > &collection)

  Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual int **size** () **const**

  Returns the number of elements in this collection.
- virtual std::vector< E > **toArray** () **const**

  Returns an array containing all of the elements in this collection.
- virtual **decaf::util::Iterator**
  < E > ∗ **iterator** ()
- virtual **decaf::util::Iterator**
  < E > ∗ **iterator** () **const**
- virtual **ListIterator**< E > ∗ **listIterator** ()
- virtual **ListIterator**< E > ∗ **listIterator** () **const**
- virtual **ListIterator**< E > ∗ **listIterator** (int index)
- virtual **ListIterator**< E > ∗ **listIterator** (int index) **const**
- virtual int **indexOf** (**const** E &value) **const**

  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 int **lastIndexOf** (**const** E &value) **const**

  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** (int index) **const**

  Gets the element contained at position passed.
- virtual E **set** (int index, **const** E &element)

  Replaces the element at the specified position in this list with the specified element.
- virtual void **add** (int index, **const** E &element)

  Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (int index, **const Collection**< E > &collection)

  Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual E **removeAt** (int index)

  Removes the element at the specified position in this list.
- virtual std::string **toString** () **const**
- bool **addIfAbsent** (**const** E &value)

  *Adds the given value to the end of this **List** (p. 1286) if it is not already contained in this **List** (p. 1286).*
- int **addAllAbsent** (**const Collection**< E > &collection)

*Every element in the given collection that is not already contained in this **Collection** (p. 660) is added to the end of this collection.*

- int **lastIndexOf** (**const** E &value, int index)

  *Searches backwards through the **List** (p. 1286) for the given element starting at the index specified.*

- int **indexOf** (**const** E &value, int index) **const**

  *Searches the **List** (p. 1286) starting from the specified index and returns the index of the first item in the list that is equal to the given value.*

- virtual void **lock** ()

  Locks the object.

- virtual bool **tryLock** ()

  *Attempts to **Lock** (p. 1304) the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** ()

  Unlocks the object.

- virtual void **wait** ()

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

- virtual void **wait** (long long millisecs)

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

- virtual void **wait** (long long millisecs, int nanos)

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

- virtual void **notify** ()

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

- virtual void **notifyAll** ()

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

Friends

- class **CopyOnWriteArraySet**

template<typename E> class decaf::util::concurrent::CopyOnWriteArrayList< E >

6.170.1 Constructor & Destructor Documentation

6.170.1.1 template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList ( ) [inline]

6.170.1.2 template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList (const Collection< E > & *collection*) [inline]

6.170.1.3 template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList (const CopyOnWriteArrayList< E > & *collection*) [inline]

6.170.1.4 template<typename E> decaf::util::concurrent::CopyOnWriteArrayList< E >::CopyOnWriteArrayList (const E * *array,* int *size*) [inline]

6.170.1.5 template<typename E> virtual decaf::util::concurrent::CopyOnWriteArrayList< E >::~CopyOnWriteArrayList ( ) [inline, virtual]

6.170.2 Member Function Documentation

6.170.2.1 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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. 662).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent().

6.170.2.2 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::add (int *index*, const E & *element*) [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	The index at which the specified element is to be inserted.
element	The element to be inserted in this **List** (p. 1286).

Exceptions

IndexOutOfBounds-Exception	if the index is greater than size of the **List** (p. 1286).
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List**< **E** > (p. 1287).

6.170.2.3 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::addAll (const Collection< E > & *collection*) [inline, 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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection< E >** (p. 663).

6.170.2.4 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::addAll (int *index*, const Collection< E > & *source*) [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. 660) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List< E >** (p. 1288).

6.170.2.5 template<typename E> int decaf::util::concurrent::CopyOnWriteArrayList< E >::addAllAbsent (const Collection< E > & *collection*) [inline]

Every element in the given collection that is not already contained in this **Collection** (p. 660) is added to the end of this collection.

The order that the elements are added is ditacted by the order that the collection's iterator returns them.

Parameters

collection	The collection whose elements are to be added if not already in this **List** (p. 1286).

Returns

the number of elements that are added to this **List** (p. 1286).

6.170.2.6 template<typename E> bool decaf::util::concurrent::CopyOnWriteArrayList< E >::addIfAbsent (const E & *value*) [inline]

Adds the given value to the end of this **List** (p. 1286) if it is not already contained in this **List** (p. 1286).

Parameters

value	The element to be added if not already contained in this **List** (p. 1286).

Returns

true if the element is added to this **List** (p. 1286).

6.170.2.7 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::clear ( ) [inline, 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

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

Implements **decaf::util::Collection< E >** (p. 663).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::operator=().

6.170.2.8 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::contains (const E & *value*) const [inline, 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 (value == NULL ? e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

Implements **decaf::util::Collection< E >** (p. 664).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::containsAll().

6.170.2.9 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::containsAll (const Collection< E > & *collection*) const [inline, virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	The **Collection** (p. 660) to compare to this one.

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

Implements **decaf::util::Collection< E >** (p. 665).

6.170.2.10 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::copy (const Collection< E > & *collection*) [inline, virtual]

Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).

Parameters

collection	The collection to mirror.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
IllegalStateException	if the elements cannot be added at this time due to insertion restrictions.

Implements **decaf::util::Collection< E >** (p. 666).

6.170.2.11 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::equals (const Collection< E > & *value*) const [inline, 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.

Implements **decaf::util::Collection< E >** (p. 666).

6.170.2.12 template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::get (int *index*) const [inline, virtual]

Gets the element contained at position passed.

Parameters

index	The position to get.

Returns

value at index specified.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.

Implements **decaf::util::List< E >** (p. 1289).

6.170.2.13 template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::indexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implements **decaf::util::List< E >** (p. 1290).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::remove().

6.170.2.14 template<typename E> int decaf::util::concurrent::CopyOnWriteArrayList< E >::indexOf (const E & *value,* int *index*) const [inline]

Searches the **List** (p. 1286) starting from the specified index and returns the index of the first item in the list that is equal to the given value.

Parameters

value	The value to search for in the **List** (p. 1286).
index	The index in the **List** (p. 1286) to begin the search from.

Returns

the index in the **List** (p. 1286) that matches the given element or -1 if not found.

Exceptions

IndexOutOfBounds-Exception	if the given index is negative.

6.170.2.15 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::isEmpty ( ) const [inline, virtual]

Returns

true if this collection contains no elements.

Implements **decaf::util::Collection< E >** (p. 667).

6.170.2.16 template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1207).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals().

6.170.2.17 template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::iterator ( ) const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1208).

6.170.2.18 template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< E >::lastIndexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implements **decaf::util::List< E >** (p. 1290).

6.170.2.19 template<typename E> int decaf::util::concurrent::CopyOnWriteArrayList< E >::lastIndexOf (const E & *value*, int *index*) [inline]

Searches backwards through the **List** (p. 1286) for the given element starting at the index specified.

Parameters

value	The value to search for in the **List** (p. 1286).
index	The index in the list to begin the search from.

Returns

the index in the list that matches the value given, or -1 if not found.

Exceptions

IndexOutOfBounds-Exception	if the given index is greater than or equal to the **List** (p. 1286) size.

6.170.2.20 template<typename E> virtual ListIterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::listIterator ( ) [inline, virtual]

Returns

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 1291).

6.170.2.21 template<typename E> virtual ListIterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::listIterator ( ) const [inline, virtual]

Implements **decaf::util::List< E >** (p. 1291).

6.170.2.22 template<typename E> virtual ListIterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::listIterator (int *index*) [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

IndexOutOfBounds-Exception	if the index is out of range (index < 0 || index > **size()** (p. 811))

Implements **decaf::util::List< E >** (p. 1292).

6.170.2.23 template<typename E> virtual ListIterator<E>* decaf::util::concurrent::CopyOnWriteArrayList< E >::listIterator (int index) const [inline, virtual]

Implements **decaf::util::List< E >** (p. 1293).

6.170.2.24 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::lock ( ) [inline, virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

6.170.2.25 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::notify ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.
RuntimeException	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2047).

6.170.2.26 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::notifyAll ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.
RuntimeException	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2048).

6.170.2.27 template<typename E> CopyOnWriteArrayList<E>& decaf::util::concurrent::CopyOnWriteArrayList< E >::operator= (const CopyOnWriteArrayList< E > & *list*) [inline]

6.170.2.28 template<typename E> CopyOnWriteArrayList<E>& decaf::util::concurrent::CopyOnWriteArrayList< E >::operator= (const Collection< E > & *list*) [inline]

6.170.2.29 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implements **decaf::util::Collection< E >** (p. 667).

6.170.2.30 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::removeAll (const Collection< E > & *collection*) [inline, 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. 660) whose elements are to be removed from this one.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implements **decaf::util::Collection< E >** (p. 668).

6.170.2.31 template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::removeAt (int *index*) [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

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

Implements **decaf::util::List< E >** (p. 1293).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::remove().

6.170.2.32 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::retainAll (const Collection< E > & *collection*) [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.

Parameters

collection	The **Collection** (p. 660) whose elements are to be retained.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implements **decaf::util::Collection< E >** (p. 669).

6.170.2.33 template<typename E> virtual E decaf::util::concurrent::CopyOnWriteArrayList< E >::set (int *index*, const E & *element*) [inline, virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters

index	The index of the element to replace.
element	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List< E >** (p. 1294).

6.170.2.34 template<typename E> virtual int decaf::util::concurrent::CopyOnWriteArrayList< 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. 669).

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::CopyOnWriteArrayList(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set().

6.170.2.35 template<typename E> virtual std::vector<E> decaf::util::concurrent::CopyOnWriteArrayList< E >::toArray ( ) const [inline, 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 in the form of an STL vector.

Implements **decaf::util::Collection< E >** (p. 670).

6.170.2.36 template<typename E> virtual std::string decaf::util::concurrent::CopyOnWriteArrayList< E >::toString ( ) const [inline, virtual]

6.170.2.37 template<typename E> virtual bool decaf::util::concurrent::CopyOnWriteArrayList< E >::tryLock ( ) [inline, virtual]

Attempts to **Lock** (p. 1304) 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. 2048).

6.170.2.38 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::unlock ( ) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

6.170.2.39 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.170.2.40 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.170.2.41 template<typename E> virtual void decaf::util::concurrent::CopyOnWriteArrayList< E >::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

6.170.3 Friends And Related Function Documentation

6.170.3.1 template<typename E> friend class CopyOnWriteArraySet [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CopyOnWriteArrayList.h**

6.171 decaf::util::concurrent::CopyOnWriteArraySet< E > Class Template Reference

Since the **CopyOnWriteArraySet** (p. 813) and the **CopyOnWriteArrayList** (p. 798) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 798) for all its underlying operations.

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

Inheritance diagram for decaf::util::concurrent::CopyOnWriteArraySet< E >:

Public Member Functions

- **CopyOnWriteArraySet** ()
- **CopyOnWriteArraySet** (**const Collection**< E > &collection)
- **CopyOnWriteArraySet** (**const** E *array, int **size**)
- virtual ~**CopyOnWriteArraySet** ()
- virtual void **copy** (**const Collection**< E > &collection)

  *Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).*
- virtual **decaf::util::Iterator** < E > * **iterator** ()

- virtual **decaf::util::Iterator** < E > * **iterator** () **const**
- virtual int **size** () **const**

  Returns the number of elements in this collection.
- virtual bool **isEmpty** () **const**

  Returns true if this collection contains no elements.
- virtual bool **add** (**const** E &value)

  Returns true if this collection changed as a result of the call.
- virtual bool **addAll** (**const Collection**< E > &collection)

  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	*The **Collection** (p. 660) whose elements are added to this one.*

  Returns

  true if this collection changed as a result of the call

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
IllegalArgumentException	*if some property of an element prevents it from being added to this collection*
IllegalStateException	*if an element cannot be added at this time due to insertion restrictions.*

  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).
- virtual void **clear** ()

  Removes all of the elements from this collection (optional operation).
- virtual bool **contains** (**const** E &value) **const**

  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 (value == NULL ? e == NULL : value == e).

  Parameters

value	*The value to check for presence in the collection.*

  Returns

  true if there is at least one of the elements in the collection

  Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

  This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.
- virtual bool **containsAll** (**const Collection**< E > &collection) **const**

  Returns true if this collection contains all of the elements in the specified collection.

  Parameters

collection	*The **Collection** (p. 660) to compare to this one.*

  Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

  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.

- virtual bool **remove** (**const** E &value)

  Removes a single instance of the specified element from the collection.
  More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this **Collection** (p. 660).*

  Returns

  true if the collection was changed, false otherwise.

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

  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.

- virtual bool **removeAll** (**const Collection**< E > &collection)

  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. 660) whose elements are to be removed from this one.*

  Returns

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

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

  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.
  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.

- virtual bool **retainAll** (**const Collection**< E > &collection)

  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. 660) whose elements are to be retained.*

  Returns

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

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

  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.

- virtual std::vector< E > **toArray** () **const**

  Answers an STL vector containing copies of all elements contained in this ***Collection*** *(*p. 660*).*

- virtual bool **equals** (**const Collection**< E > &collection) **const**

  Answers true if this ***Collection*** *(*p. 660*) and the one given are the same size and if each element contained in the* ***Collection*** *(*p. 660*) given is equal to an element contained in this collection.*

6.171.1 Detailed Description

template<typename E>class decaf::util::concurrent::CopyOnWriteArraySet< E >

Since the **CopyOnWriteArraySet** (p. 813) and the **CopyOnWriteArrayList** (p. 798) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 798) for all its underlying operations.

This collection is best used in applications where the **Set** (p. 1857) size is usually small and write operations are minimal as they result in a copy of the underlying array being created. Reads are generally fast and the iterators provided by this collection do not block as they operate on a snapshot of the data taken at the time of their creation.

Since

1.0

6.171.2 Constructor & Destructor Documentation

6.171.2.1 template<typename E > decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet ( ) [inline]

6.171.2.2 template<typename E > decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet (const Collection< E > & *collection*) [inline]

References decaf::util::concurrent::CopyOnWriteArraySet< E >::copy().

6.171.2.3 template<typename E > decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet (const E * *array*, int *size*) [inline]

References decaf::util::concurrent::CopyOnWriteArraySet< E >::size().

6.171.2.4 template<typename E > virtual decaf::util::concurrent::CopyOnWriteArraySet< E >::~CopyOnWriteArraySet ( ) [inline, virtual]

6.171.3 Member Function Documentation

6.171.3.1 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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. 662).

6.171.3.2 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::addAll (const Collection< E > & *collection*) [inline, 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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

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).

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).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 87).

6.171.3.3 template<typename E > virtual void decaf::util::concurrent::CopyOnWriteArraySet< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 88).

6.171.3.4 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::contains (const E & *value*) const [inline, 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 (value == NULL ? e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 88).

6.171.3.5 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::containsAll (const Collection< E > & *collection*) const [inline, virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	The **Collection** (p. 660) to compare to this one.

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 89).

6.171.3.6 template<typename E > virtual void decaf::util::concurrent::CopyOnWriteArraySet< E >::copy (const Collection< E > & *collection*) [inline, virtual]

Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 660)'s clear method.

Parameters

collection	The collection to mirror.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
IllegalStateException	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 89).

Referenced by decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet().

6.171.3.7 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::equals (const Collection< E > & *collection*) const [inline, virtual]

Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.

Parameters

collection	- The **Collection** (p. 660) to be compared to this one.

Returns

true if this **Collection** (p. 660) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

References decaf::lang::Iterable< E >::iterator(), decaf::util::concurrent::CopyOnWriteArraySet< E >::size(), and decaf::util::Collection< E >::size().

6.171.3.8 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::isEmpty ( ) const [inline, virtual]

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 822) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

6.171.3.9 template<typename E > virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArraySet< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1207).

6.171.3.10 template<typename E > virtual decaf::util::Iterator<E>* decaf::util::concurrent::CopyOnWriteArraySet< E >::iterator ( ) const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1208).

6.171.3.11 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 92).

6.171.3.12 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::removeAll (const Collection< E > & *collection*) [inline, 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. 660) whose elements are to be removed from this one.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

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.

Reimplemented from **decaf::util::AbstractSet< E >** (p. 119).

6.171.3.13 template<typename E > virtual bool decaf::util::concurrent::CopyOnWriteArraySet< E >::retainAll (const Collection< E > & *collection*) [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.

Parameters

collection	The **Collection** (p. 660) whose elements are to be retained.

Returns

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

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 93).

6.171.3.14 template<typename E > virtual int decaf::util::concurrent::CopyOnWriteArraySet< 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. 669).

Referenced by decaf::util::concurrent::CopyOnWriteArraySet< E >::CopyOnWriteArraySet(), and decaf::util::concurrent::CopyOnWriteArraySet< E >::equals().

6.171.3.15 template<typename E > virtual std::vector<E> decaf::util::concurrent::CopyOnWriteArraySet< E >::toArray ( ) const [inline, virtual]

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).

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. 660)

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 94).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CopyOnWriteArraySet.h**

6.172 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** ()

  Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.
- virtual bool **await** (long long timeOut)

  Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.
- virtual bool **await** (long long timeout, **const TimeUnit** &unit)

  Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.
- virtual void **countDown** ()

  Counts down the latch, releasing all waiting threads when the count hits zero.
- virtual int **getCount** () **const**

  Gets the current count.

6.172.1 Constructor & Destructor Documentation

6.172.1.1 decaf::util::concurrent::CountDownLatch::CountDownLatch (int *count*)

Constructor.

Parameters

count	- number to count down from.

6.172.1.2 virtual decaf::util::concurrent::CountDownLatch::~CountDownLatch ( ) [virtual]

6.172.2 Member Function Documentation

6.172.2.1 virtual void decaf::util::concurrent::CountDownLatch::await ( ) [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.

If the current count is zero then this method returns immediately.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happen:

- The count reaches zero due to invocations of the **countDown()** (p. 825) method; 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,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Exceptions

InterruptedException	- if the current thread is interrupted while waiting.
Exception	- if any other error occurs.

6.172.2.2 virtual bool decaf::util::concurrent::CountDownLatch::await (long long *timeOut*) [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

- The count reaches zero due to invocations of the **countDown()** (p. 825) method; or
  - Some other thread interrupts the current thread; or
  - The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

- has its interrupted status set on entry to this method; or
  - is interrupted while waiting,

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	- Time in milliseconds to wait for the count to reach zero.

Exceptions

InterruptedException	- if the current thread is interrupted while waiting.
Exception	- if any other error occurs.

6.172.2.3 virtual bool decaf::util::concurrent::CountDownLatch::await (long long *timeout,* const TimeUnit & *unit*) [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

- The count reaches zero due to invocations of the **countDown()** (p. 825) method; or
  - Some other thread interrupts the current thread; or
  - The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

- has its interrupted status set on entry to this method; or
  - is interrupted while waiting,

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	- Time to wait for the count to reach zero.
unit	- The units that the timeout specifies.

Exceptions

InterruptedException	- if the current thread is interrupted while waiting.
Exception	- if any other error occurs.

6.172.2.4 virtual void decaf::util::concurrent::CountDownLatch::countDown ( ) [virtual]

Counts down the latch, releasing all waiting threads when the count hits zero.

6.172.2.5 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**

6.173 decaf::util::zip::CRC32 Class Reference

Class that can be used to compute a CRC-32 checksum for a data stream.

`#include <src/main/decaf/util/zip/CRC32.h>`

Inheritance diagram for decaf::util::zip::CRC32:

Public Member Functions

- **CRC32** ()
- virtual ~**CRC32** ()
- virtual long long **getValue** () **const**
- virtual void **reset** ()

  Reset the checksum to its initial value.
- virtual void **update** (**const** std::vector< unsigned char > &buffer)

  Updates the current checksum with the specified vector of bytes.
- virtual void **update** (**const** std::vector< unsigned char > &buffer, int offset, int length)

  Updates the current checksum with the specified array of bytes.
- virtual void **update** (**const** unsigned char *buffer, int size, int offset, int length)

  Updates the current checksum with the specified array of bytes.
- virtual void **update** (int byte)

  Updates the current checksum with the specified byte value.

6.173.1 Detailed Description

Class that can be used to compute a CRC-32 checksum for a data stream.

Since

1.0

6.173.2 Constructor & Destructor Documentation

6.173.2.1 decaf::util::zip::CRC32::CRC32 ( )

6.173.2.2 virtual decaf::util::zip::CRC32::~CRC32 ( ) [virtual]

6.173.3 Member Function Documentation

6.173.3.1 virtual long long decaf::util::zip::CRC32::getValue ( ) const [virtual]

Returns

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 629).

6.173.3.2 virtual void decaf::util::zip::CRC32::reset ( ) [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 629).

6.173.3.3 virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & *buffer*) [virtual]

Updates the current checksum with the specified vector of bytes.

Parameters

buffer	The buffer to read the updated bytes from.

Implements **decaf::util::zip::Checksum** (p. 629).

6.173.3.4 virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & *buffer,* int *offset,* int *length*) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer	The buffer to read the updated bytes from.
offset	The position in the buffer to start reading.
length	The amount of data to read from the byte buffer.

Exceptions

IndexOutOfBounds-Exception	if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 629).

6.173.3.5 virtual void decaf::util::zip::CRC32::update (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [virtual]

Updates the current checksum with the specified array of bytes.

Parameters

buffer	The buffer to read the updated bytes from.
size	The size of the passed buffer.
offset	The position in the buffer to start reading.
length	The amount of data to read from the byte buffer.

Exceptions

NullPointerException	if the passed buffer is NULL.
IndexOutOfBounds-Exception	if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 629).

6.173.3.6 virtual void decaf::util::zip::CRC32::update (int *byte*) [virtual]

Updates the current checksum with the specified byte value.

Parameters

byte	The byte value to update the current **Checksum** (p. 628) with (0..255).

Implements **decaf::util::zip::Checksum** (p. 630).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CRC32.h**

6.174 ct_data_s Struct Reference

`#include <src/main/decaf/internal/util/zip/deflate.h>`

Data Fields

- union {
  ush freq
  ush code
  } **fc**

- union {
  ush dad
  ush len
  } **dl**

6.174.1 Field Documentation

6.174.1.1 ush ct_data_s::code

6.174.1.2 ush ct_data_s::dad

6.174.1.3 union { ... } ct_data_s::dl

6.174.1.4 union { ... } ct_data_s::fc

6.174.1.5 ush ct_data_s::freq

6.174.1.6 ush ct_data_s::len

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**deflate.h**

6.175 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **DataArrayResponse** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 Attributes

- std::vector < **decaf::lang::Pointer** < **DataStructure** > > **data**

6.175.1 Constructor & Destructor Documentation

6.175.1.1 activemq::commands::DataArrayResponse::DataArrayResponse ( )

6.175.1.2 virtual activemq::commands::DataArrayResponse::~DataArrayResponse ( ) [virtual]

6.175.2 Member Function Documentation

6.175.2.1 virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1782).

6.175.2.2 virtual void activemq::commands::DataArrayResponse::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::Response** (p. 1782).

6.175.2.3 virtual bool activemq::commands::DataArrayResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.175.2.4 virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData ( ) const [virtual]

6.175.2.5 virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData ( ) [virtual]

6.175.2.6 virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 1783).

6.175.2.7 virtual void activemq::commands::DataArrayResponse::setData (const std::vector< decaf::lang::Pointer< DataStructure > > & *data*) [virtual]

6.175.2.8 virtual std::string activemq::commands::DataArrayResponse::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.175.3 Field Documentation

6.175.3.1 std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data [protected]

6.175.3.2 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**

6.176 activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 831).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/DataArrayResponseMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual ~**DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.176.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 831).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.176.2 Constructor & Destructor Documentation

6.176.2.1 activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::DataArrayResponseMarshaller ( ) [inline]

6.176.2.2 virtual activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::~DataArrayResponseMarshaller ( ) [inline, virtual]

6.176.3 Member Function Documentation

6.176.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1789).

6.176.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.176.3.3 virtual void activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.176.3.4 virtual void activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.176.3.5 virtual int activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.176.3.6 virtual void activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.176.3.7 virtual void activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**DataArrayResponseMarshaller.h**

6.177 decaf::util::zip::DataFormatException Class Reference

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

Inheritance diagram for decaf::util::zip::DataFormatException:

Public Member Functions

- **DataFormatException** () throw ()
  Default Constructor.
- **DataFormatException** (**const lang::Exception** &ex) throw ()
  Copy Constructor.
- **DataFormatException** (**const DataFormatException** &ex) throw ()
  Copy Constructor.
- **DataFormatException** (**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.
- **DataFormatException** (**const** std::exception ∗**cause**) throw ()
  Constructor.
- **DataFormatException** (**const** char ∗file, **const** int lineNumber, **const** char ∗msg,...) throw ()
  Constructor.
- virtual **DataFormatException** ∗ **clone** () **const**
  Clones this exception.
- virtual ∼**DataFormatException** () throw ()

6.177.1 Constructor & Destructor Documentation

6.177.1.1 decaf::util::zip::DataFormatException::DataFormatException ( ) throw ()

Default Constructor.

6.177.1.2 decaf::util::zip::DataFormatException::DataFormatException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.177.1.3 decaf::util::zip::DataFormatException::DataFormatException (const DataFormatException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.177.1.4 decaf::util::zip::DataFormatException::DataFormatException (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

6.177.1.5 decaf::util::zip::DataFormatException::DataFormatException (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.

6.177.1.6 decaf::util::zip::DataFormatException::DataFormatException (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

6.177.1.7 virtual decaf::util::zip::DataFormatException::~DataFormatException ( ) throw () [virtual]

6.177.2 Member Function Documentation

6.177.2.1 virtual DataFormatException* decaf::util::zip::DataFormatException::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. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**DataFormatException.h**

6.178 decaf::net::DatagramPacket Class Reference

Class that represents a single datagram packet.

`#include <src/main/decaf/net/DatagramPacket.h>`

Public Member Functions

- **DatagramPacket** (unsigned char *bytes, int size, int length)

  *Creates a new **DatagramPacket** (p. 836) for use in receiving a packet of the given length.*
- **DatagramPacket** (unsigned char *bytes, int size, int offset, int length)

  *Creates a new **DatagramPacket** (p. 836) for use in receiving a packet of the given length starting at the specified offset into the buffer.*
- **DatagramPacket** (unsigned char *bytes, int size, int offset, int length, **const InetAddress** &address, int port)

  *Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified host on the specified port.*
- **DatagramPacket** (unsigned char *bytes, int size, int length, **const InetAddress** &address, int port)

  *Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length to the specified host on the specified port.*
- **DatagramPacket** (unsigned char *bytes, int size, int length, **const SocketAddress** &address)

  *Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length into the buffer to the specified socket address.*
- **DatagramPacket** (unsigned char *bytes, int size, int offset, int length, **const SocketAddress** &address)

  *Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified socket address.*
- virtual ~**DatagramPacket** ()
- **InetAddress** * **getAddress** () **const**
- void **setAddress** (**const InetAddress** &address)

  Sets the IP address of the machine to which this datagram is being sent.
- **SocketAddress** * **getSocketAddress** () **const**

  *Gets the **SocketAddress** (p. 1913) (usually IP address + port number) of the remote host that this packet is being sent to or is coming from.*
- void **setSocketAddress** (**const SocketAddress** &address)

  *Sets the **SocketAddress** (p. 1913) (usually IP address + port number) of the remote host to which this datagram is being sent.*
- int **getPort** () **const**
- void **setPort** (int port)

  Sets the port number on the remote host to which this datagram is being sent.
- int **getOffset** () **const**
- void **setOffset** (int offset)

Sets the offset into the data buffer where data to sent is to be read from or where the data that is received should start writing to.

- int **getLength** () **const**
- void **setLength** (int length)

  Set the length for this packet.
- unsigned char * **getData** () **const**
- int **getSize** () **const**
- void **setData** (unsigned char *buffer, int size)

  Set the data buffer for this packet.
- void **setData** (unsigned char *buffer, int size, int offset, int length)

  Set the data buffer for this packet.

6.178.1 Detailed Description

Class that represents a single datagram packet.

Datagrams are sent in packets from machine to machine and can each be routed differently and can arrive in any order. Delivery of a packet is not guaranteed.

Since

1.0

6.178.2 Constructor & Destructor Documentation

6.178.2.1 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *length*)

Creates a new **DatagramPacket** (p. 836) for use in receiving a packet of the given length.

Parameters

bytes	The array of bytes to hold the incoming datagram buffer.
size	The size of the supplied byte array.
length	The number of byte to read starting at the supplied offset.

Exceptions

NullPointerException	if the pointer to the buffer is NULL.
IndexOutOfBounds-Exception	if the number of bytes to read exceeds the buffer size.

6.178.2.2 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *offset*, int *length*)

Creates a new **DatagramPacket** (p. 836) for use in receiving a packet of the given length starting at the specified offset into the buffer.

Parameters

bytes	The array of bytes to hold the incoming datagram buffer.
size	The size of the supplied byte array.
offset	The position in the array to start writing to.
length	The number of byte to read starting at the supplied offset.

Exceptions

NullPointerException	if the pointer to the buffer is NULL.
IndexOutOfBounds-Exception	if the number of bytes to copy exceeds the buffer size.

6.178.2.3 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *offset*, int *length*, const InetAddress & *address*, int *port*)

Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified host on the specified port.

Parameters

bytes	The array of bytes to hold the outgoing datagram buffer.
size	The size of the supplied byte array.
offset	The position in the array to start writing to.
length	The number of byte to read starting at the supplied offset.
address	The Address to send the packet to
port	The port on the destination that is to receive this packet.

Exceptions

NullPointerException	if the pointer to the buffer is NULL.
IndexOutOfBounds-Exception	if the number of bytes to copy exceeds the buffer size.

6.178.2.4 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *length*, const InetAddress & *address*, int *port*)

Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length to the specified host on the specified port.

Parameters

bytes	The array of bytes to hold the outgoing datagram buffer.
size	The size of the supplied byte array.
length	The number of byte to read starting at the supplied offset.
address	The Address to send the packet to
port	The port on the destination that is to receive this packet.

Exceptions

NullPointerException	if the pointer to the buffer is NULL.
IndexOutOfBounds-Exception	if the number of bytes to copy exceeds the buffer size.

6.178.2.5 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *length*, const SocketAddress & *address*)

Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length into the buffer to the specified socket address.

Parameters

bytes	The array of bytes to hold the outgoing datagram buffer.
size	The size of the supplied byte array.
length	The number of byte to read starting at the supplied offset.
address	The Address to send the packet to

Exceptions

NullPointerException	if the pointer to the buffer is NULL.
IndexOutOfBounds-Exception	if the number of bytes to copy exceeds the buffer size.

6.178.2.6 decaf::net::DatagramPacket::DatagramPacket (unsigned char * *bytes*, int *size*, int *offset*, int *length*, const SocketAddress & *address*)

Creates a new **DatagramPacket** (p. 836) for use in sending a packet of the given length starting at the specified offset into the buffer to the specified socket address.

Parameters

bytes	The array of bytes to hold the outgoing datagram buffer.
size	The size of the supplied byte array.
offset	The position in the array to start writing to.
length	The number of byte to read starting at the supplied offset.
address	The Address to send the packet to

Exceptions

NullPointerException	if the pointer to the buffer is NULL.
IndexOutOfBounds-Exception	if the number of bytes to copy exceeds the buffer size.

6.178.2.7 virtual decaf::net::DatagramPacket::~DatagramPacket ( ) [virtual]

6.178.3 Member Function Documentation

6.178.3.1 InetAddress* decaf::net::DatagramPacket::getAddress ( ) const

Returns

the IP address that this datagram packet is being sent to or was received from.

6.178.3.2 unsigned char* decaf::net::DatagramPacket::getData ( ) const

Returns

the data buffer. The data received or the data to be sent starts from the offset in the buffer, and continues for length bytes.

6.178.3.3 int decaf::net::DatagramPacket::getLength ( ) const

Returns

the length of the data to be sent or the length of the data received.

6.178.3.4 int decaf::net::DatagramPacket::getOffset ( ) const

Returns

the offset of the data to be sent or the offset of the data received.

6.178.3.5 int decaf::net::DatagramPacket::getPort ( ) const

Returns

the port number that this datagram packet is being sent to or was received from.

6.178.3.6 int decaf::net::DatagramPacket::getSize ( ) const

Returns

the size of the buffer used in this datagram packet.

6.178.3.7 SocketAddress* decaf::net::DatagramPacket::getSocketAddress ( ) const

Gets the **SocketAddress** (p. 1913) (usually IP address + port number) of the remote host that this packet is being sent to or is coming from.

Returns

the **SocketAddress** (p. 1913) for this datagram packet.

6.178.3.8 void decaf::net::DatagramPacket::setAddress (const InetAddress & *address*)

Sets the IP address of the machine to which this datagram is being sent.

Parameters

address	The IP address.

6.178.3.9 void decaf::net::DatagramPacket::setData (unsigned char * *buffer*, int *size*)

Set the data buffer for this packet.

With the offset of this **DatagramPacket** (p. 836) set to 0, and the length set to the size value specified.

Parameters

buffer	The new data buffer to use for this datagram packet.
size	The size of the buffer.

Exceptions

NullPointerException	if the buffer pointer is NULL.

6.178.3.10 void decaf::net::DatagramPacket::setData (unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Set the data buffer for this packet.

With the offset of this **DatagramPacket** (p. 836) set to 0, and the length set to the size value specified.

Parameters

buffer	The new data buffer to use for this datagram packet.
size	The size of the buffer.
offset	The position in the buffer to read from or write to.
length	The number of bytes that will be read into the buffer or sent from the buffer.

Exceptions

NullPointerException	if the buffer pointer is NULL.

6.178.3.11 void decaf::net::DatagramPacket::setLength (int *length*)

Set the length for this packet.

The length of the packet is the number of bytes from the packet's data buffer that will be sent, or the number of bytes of the packet's data buffer that will be used for receiving data. The length must be lesser or equal to the offset plus the length of the packet's buffer.

Parameters

length	The length value to set for this packet.

Exceptions

IllegalArgumentException	if the value is negative or exceeds the data buffers length.

6.178.3.12 void decaf::net::DatagramPacket::setOffset (int *offset*)

Sets the offset into the data buffer where data to sent is to be read from or where the data that is received should start writing to.

Parameters

offset	The buffer offset value.

Exceptions

IllegalArgumentException	if the offset value is greater than the buffer size.

6.178.3.13 void decaf::net::DatagramPacket::setPort (int *port*)

Sets the port number on the remote host to which this datagram is being sent.

Parameters

port	The port on the remote host.

Exceptions

IllegalArgumentException	if the port value is not in the range [0..65535].

6.178.3.14 void decaf::net::DatagramPacket::setSocketAddress (const SocketAddress & *address*)

Sets the **SocketAddress** (p. 1913) (usually IP address + port number) of the remote host to which this datagram is being sent.

Parameters

address	The **SocketAddress** (p. 1913) (IP + port) for this datagram packet.

Exceptions

IllegalArgumentException	if the subclass of address is not supported by this **Socket** (p. 1900).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**DatagramPacket.h**

6.179 decaf::io::DataInput Class Reference

The **DataInput** (p. 842) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

`#include <src/main/decaf/io/DataInput.h>`

Public Member Functions

- virtual ~**DataInput** ()
- virtual bool **readBoolean** ()=0

  Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.
- virtual char **readByte** ()=0

  Reads and returns one input byte.
- virtual unsigned char **readUnsignedByte** ()=0

  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** ()=0

  Reads an input char and returns the char value.
- virtual double **readDouble** ()=0

  Reads eight input bytes and returns a double value.
- virtual float **readFloat** ()=0

  Reads four input bytes and returns a float value.
- virtual int **readInt** ()=0

  Reads four input bytes and returns an int value.
- virtual long long **readLong** ()=0

  Reads eight input bytes and returns a long value.
- virtual short **readShort** ()=0

  Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** ()=0

  Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** ()=0

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

- virtual std::string **readLine** ()=0

  Reads the next line of text from the input stream.

- virtual std::string **readUTF** ()=0

  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 UTFFormat-Exception.

- virtual void **readFully** (unsigned char *buffer, int size)=0

  Reads some bytes from an input stream and stores them into the buffer array buffer.

- virtual void **readFully** (unsigned char *buffer, int size, int offset, int length)=0

  Reads length bytes from an input stream.

- virtual long long **skipBytes** (long long num)=0

  Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.179.1 Detailed Description

The **DataInput** (p. 842) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

There is also a facility for reconstructing Strings from data in the Java standard modified UTF-8 format.

It is generally true of all the reading routines in this interface that if end of file is reached before the desired number of bytes has been read, an **EOFException** (p. 986) is thrown. If any byte cannot be read for any reason other than end of file, an **IOException** (p. 1198) other than **EOFException** (p. 986) is thrown. for example, an **IOException** (p. 1198) may be thrown if the underlying input stream has been closed.

See also

DataOutput (p. 856)
DataInputStream (p. 849)

Since

1.0

6.179.2 Constructor & Destructor Documentation

6.179.2.1 virtual decaf::io::DataInput::~DataInput ( ) [inline, virtual]

6.179.3 Member Function Documentation

6.179.3.1 virtual bool decaf::io::DataInput::readBoolean ( ) [pure virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value of the read in byte (0=false, 1=true).

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.2 virtual char decaf::io::DataInput::readByte ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.3 virtual char decaf::io::DataInput::readChar ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.4 virtual double decaf::io::DataInput::readDouble ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.5 virtual float decaf::io::DataInput::readFloat ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.6 virtual void decaf::io::DataInput::readFully (unsigned char * *buffer,* int *size*) [pure 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's 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. 986) is thrown.
- An I/O error occurs, in which case an **IOException** (p. 1198) other than **EOFException** (p. 986) 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	The byte array to insert read data into.
size	The size in bytes of the given byte buffer.

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.
IndexOutOfBounds-Exception	if the size value is negative.

6.179.3.7 virtual void decaf::io::DataInput::readFully (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [pure 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. 986) is thrown.
- An I/O error occurs, in which case an **IOException** (p. 1198) other than **EOFException** (p. 986) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length 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 length.

Parameters

buffer	The byte array to insert read data into.
size	The size in bytes of the given byte buffer.
offset	The location in buffer to start writing.
length	The number of bytes to read from the buffer.

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.
NullPointerException	if the buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length $>$ size, or an int param is negative.

6.179.3.8 virtual int decaf::io::DataInput::readInt ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.9 virtual std::string decaf::io::DataInput::readLine ( ) [pure virtual]

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character '

' is encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character '

', then that is discarded also; reading then ceases. If end of file is encountered before either of the characters '

' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

Returns

the next line of text read from the input stream or empty string if at EOF.

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.

6.179.3.10 virtual long long decaf::io::DataInput::readLong ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.11 virtual short decaf::io::DataInput::readShort ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.12 virtual std::string decaf::io::DataInput::readString ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.13 virtual unsigned char decaf::io::DataInput::readUnsignedByte ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.14 virtual unsigned short decaf::io::DataInput::readUnsignedShort ( ) [pure 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.179.3.15 virtual std::string decaf::io::DataInput::readUTF ( ) [pure 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. 860) 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.
UTFDataFormatException *(p. 2228)*	if the bytes are not valid modified UTF-8 values.

6.179.3.16 virtual long long decaf::io::DataInput::skipBytes (long long *num*) [pure 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. 986). The actual number of bytes skipped is returned.

Parameters

num	The number of bytes to skip over.

Returns

the total number of bytes skipped.

Exceptions

IOException *(*p. *1198)*	if an I/O Error occurs.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataInput.h**

6.180 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. *849) that uses the specified underlying* ***InputStream*** *(*p. *1134).*
- virtual ∼**DataInputStream** ()
- virtual bool **readBoolean** ()

  Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.
- virtual char **readByte** ()

  Reads and returns one input byte.
- virtual unsigned char **readUnsignedByte** ()

  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** ()

  Reads an input char and returns the char value.
- virtual double **readDouble** ()

  Reads eight input bytes and returns a double value.
- virtual float **readFloat** ()

  Reads four input bytes and returns a float value.
- virtual int **readInt** ()

  Reads four input bytes and returns an int value.
- virtual long long **readLong** ()

  Reads eight input bytes and returns a long value.
- virtual short **readShort** ()

  Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** ()

  Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** ()

  Reads an NULL terminated ASCII string to the stream and returns the string to the caller.
- virtual std::string **readLine** ()

  Reads the next line of text from the input stream.
- virtual std::string **readUTF** ()

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 UTFFormat-Exception.

- virtual void **readFully** (unsigned char *buffer, int size)

  Reads some bytes from an input stream and stores them into the buffer array buffer.
- virtual void **readFully** (unsigned char *buffer, int size, int offset, int length)

  Reads length bytes from an input stream.
- virtual long long **skipBytes** (long long num)

  Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.180.1 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. 1134) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

DataInputStream (p. 849) os = new **DataInputStream** (p. 849)(new **InputStream()** (p. 1135), true)

Since

1.0

6.180.2 Constructor & Destructor Documentation

6.180.2.1 decaf::io::DataInputStream::DataInputStream (InputStream * *inputStream*, bool *own* = false)

Creates a **DataInputStream** (p. 849) that uses the specified underlying **InputStream** (p. 1134).

Parameters

inputStream	the **InputStream** (p. 1134) instance to wrap.
own	indicates if this class owns the wrapped string defaults to false.

6.180.2.2 virtual decaf::io::DataInputStream::~DataInputStream ( ) [virtual]

6.180.3 Member Function Documentation

6.180.3.1 virtual bool decaf::io::DataInputStream::readBoolean ( ) [virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value of the read in byte (0=false, 1=true).

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.2 virtual char decaf::io::DataInputStream::readByte ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.3 virtual char decaf::io::DataInputStream::readChar ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.4 virtual double decaf::io::DataInputStream::readDouble ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.5 virtual float decaf::io::DataInputStream::readFloat ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.6 virtual void decaf::io::DataInputStream::readFully (unsigned char * *buffer,* int *size*) [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's 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. 986) is thrown.
- An I/O error occurs, in which case an **IOException** (p. 1198) other than **EOFException** (p. 986) 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	The byte array to insert read data into.
size	The size in bytes of the given byte buffer.

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.
IndexOutOfBounds-Exception	if the size value is negative.

6.180.3.7 virtual void decaf::io::DataInputStream::readFully (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [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. 986) is thrown.
- An I/O error occurs, in which case an **IOException** (p. 1198) other than **EOFException** (p. 986) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length 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 length.

Parameters

buffer	The byte array to insert read data into.
size	The size in bytes of the given byte buffer.
offset	The location in buffer to start writing.
length	The number of bytes to read from the buffer.

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.
NullPointerException	if the buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length $>$ size.

6.180.3.8 virtual int decaf::io::DataInputStream::readInt ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.9 virtual std::string decaf::io::DataInputStream::readLine ( ) [virtual]

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character '

' is encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character '

', then that is discarded also; reading then ceases. If end of file is encountered before either of the characters '

' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

Returns

the next line of text read from the input stream or empty string if at EOF.

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.

6.180.3.10 virtual long long decaf::io::DataInputStream::readLong ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.11 virtual short decaf::io::DataInputStream::readShort ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.12 virtual std::string decaf::io::DataInputStream::readString ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.13 virtual unsigned char decaf::io::DataInputStream::readUnsignedByte ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.14 virtual unsigned short decaf::io::DataInputStream::readUnsignedShort ( ) [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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.

6.180.3.15 virtual std::string decaf::io::DataInputStream::readUTF ( ) [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. 860) 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. 1198)*	if an I/O Error occurs.
EOFException *(p. 986)*	if the end of input is reached.
UTFDataFormatException *(p. 2228)*	if the bytes are not valid modified UTF-8 values.

6.180.3.16 virtual long long decaf::io::DataInputStream::skipBytes (long long *num*) [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. 986). The actual number of bytes skipped is returned.

Parameters

num	The number of bytes to skip over.

Returns

the total number of bytes skipped.

Exceptions

IOException *(p. 1198)*	if an I/O Error occurs.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataInputStream.h**

6.181 decaf::io::DataOutput Class Reference

The **DataOutput** (p. 856) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

`#include <src/main/decaf/io/DataOutput.h>`

Public Member Functions

- virtual ~**DataOutput** ()
- virtual void **writeBoolean** (bool value)=0

  Writes a boolean to the underlying output stream as a 1-byte value.
- virtual void **writeByte** (unsigned char value)=0

  Writes out a byte to the underlying output stream as a 1-byte value.
- virtual void **writeShort** (short value)=0

  Writes a short to the underlying output stream as two bytes, high byte first.
- virtual void **writeUnsignedShort** (unsigned short value)=0

  Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual void **writeChar** (char value)=0

  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)=0

  Writes an int to the underlying output stream as four bytes, high byte first.
- virtual void **writeLong** (long long value)=0

  Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.
- virtual void **writeFloat** (float value)=0

  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)=0

  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)=0

  Writes out the string to the underlying output stream as a sequence of bytes.
- virtual void **writeChars** (**const** std::string &value)=0

  Writes a string to the underlying output stream as a sequence of characters.
- virtual void **writeUTF** (**const** std::string &value)=0

  Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

6.181.1 Detailed Description

The **DataOutput** (p. 856) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

There is also a facility for converting Strings into the Java standard modified UTF-8 format and writing the resulting series of bytes.

If a method in this interface encounters an error while writing it will throw an **IOException** (p. 1198).

See also

DataInput (p. 842)
DataOutputStream (p. 860)

Since

1.0

6.181.2 Constructor & Destructor Documentation

6.181.2.1 virtual decaf::io::DataOutput::~DataOutput ( ) [inline, virtual]

6.181.3 Member Function Documentation

6.181.3.1 virtual void decaf::io::DataOutput::writeBoolean (bool *value*) [pure 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 as a byte (1=true, 0=false).

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.2 virtual void decaf::io::DataOutput::writeByte (unsigned char *value*) [pure 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. 1198)*	if an I/O error is encountered.

6.181.3.3 virtual void decaf::io::DataOutput::writeBytes (const std::string & *value*) [pure 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 vector of bytes to write.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.4 virtual void decaf::io::DataOutput::writeChar (char *value*) [pure 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 signed char value to write.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.5 virtual void decaf::io::DataOutput::writeChars (const std::string & *value*) [pure 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 string value to write as raw bytes.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.6 virtual void decaf::io::DataOutput::writeDouble (double *value*) [pure 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 64bit double value to write.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.7 virtual void decaf::io::DataOutput::writeFloat (float *value*) [pure 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 32bit floating point value to write.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.8 virtual void decaf::io::DataOutput::writeInt (int *value*) [pure 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 signed integer value to write.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.9 virtual void decaf::io::DataOutput::writeLong (long long *value*) [pure 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 signed 64bit long value to write.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.10 virtual void decaf::io::DataOutput::writeShort (short *value*) [pure 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 signed short value to write.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.11 virtual void decaf::io::DataOutput::writeUnsignedShort (unsigned short *value*) [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

value	The unsigned short to write to the stream.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.

6.181.3.12 virtual void decaf::io::DataOutput::writeUTF (const std::string & *value*) [pure virtual]

Writes out the string to the underlying output stream as a modeified 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 string value value to write as modified UTF-8.

Exceptions

IOException *(p. 1198)*	if an I/O error is encountered.
UTFDataFormatException *(p. 2228)*	if the encoded size if greater than 65535

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutput.h**

6.182 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 long long **size** () **const**

  Returns the current value of the counter written, the number of bytes written to this data output stream so far.
- virtual void **writeBoolean** (bool value)
- virtual void **writeByte** (unsigned char value)
- virtual void **writeShort** (short value)
- virtual void **writeUnsignedShort** (unsigned short value)
- virtual void **writeChar** (char value)
- virtual void **writeInt** (int value)
- virtual void **writeLong** (long long value)
- virtual void **writeFloat** (float value)
- virtual void **writeDouble** (double value)
- virtual void **writeBytes** (**const** std::string &value)
- virtual void **writeChars** (**const** std::string &value)
- virtual void **writeUTF** (**const** std::string &value)

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (**const** unsigned char *__buffer__, int **size**, int offset, int length)

Protected Attributes

- long long **written**
- unsigned char **buffer** [8]

6.182.1 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.

6.182.2 Constructor & Destructor Documentation

6.182.2.1 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.

6.182.2.2 virtual decaf::io::DataOutputStream::~DataOutputStream ( ) [virtual]

6.182.3 Member Function Documentation

6.182.3.1 virtual void decaf::io::DataOutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.182.3.2 virtual void decaf::io::DataOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.182.3.3 virtual long long 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 **decaf::lang::Long::MAX_VALUE** (p. 1351).

Returns

the value of the written field.

6.182.3.4 virtual void decaf::io::DataOutputStream::writeBoolean (bool *value*) [virtual]

Referenced by activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray().

6.182.3.5 virtual void decaf::io::DataOutputStream::writeByte (unsigned char *value*) [virtual]

6.182.3.6 virtual void decaf::io::DataOutputStream::writeBytes (const std::string & *value*) [virtual]

6.182.3.7 virtual void decaf::io::DataOutputStream::writeChar (char *value*) [virtual]

6.182.3.8 virtual void decaf::io::DataOutputStream::writeChars (const std::string & *value*) [virtual]

6.182.3.9 virtual void decaf::io::DataOutputStream::writeDouble (double *value*) [virtual]

6.182.3.10 virtual void decaf::io::DataOutputStream::writeFloat (float *value*) [virtual]

6.182.3.11 virtual void decaf::io::DataOutputStream::writeInt (int *value*) [virtual]

6.182.3.12 virtual void decaf::io::DataOutputStream::writeLong (long long *value*) [virtual]

6.182.3.13 virtual void decaf::io::DataOutputStream::writeShort (short *value*) [virtual]

Referenced by activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray(), and activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2().

6.182.3.14 virtual void decaf::io::DataOutputStream::writeUnsignedShort (unsigned short *value*) [virtual]

6.182.3.15 virtual void decaf::io::DataOutputStream::writeUTF (const std::string & *value*) [virtual]

6.182.4 Field Documentation

6.182.4.1 unsigned char decaf::io::DataOutputStream::buffer[8] [protected]

6.182.4.2 long long decaf::io::DataOutputStream::written [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutputStream.h**

6.183 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 **DataStructure*** (p. 877) *Type as defined in CommandTypes.h.*
- virtual **DataResponse** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) *such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure*** (p. 877) *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 Attributes

- **Pointer< DataStructure > data**

6.183.1 Constructor & Destructor Documentation

6.183.1.1 activemq::commands::DataResponse::DataResponse ( )

6.183.1.2 virtual activemq::commands::DataResponse::~DataResponse ( ) [virtual]

6.183.2 Member Function Documentation

6.183.2.1 virtual DataResponse* activemq::commands::DataResponse::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1782).

6.183.2.2 virtual void activemq::commands::DataResponse::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::Response** (p. 1782).

6.183.2.3 virtual bool activemq::commands::DataResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.183.2.4 virtual const Pointer<DataStructure>& activemq::commands::DataResponse::getData ( ) const [virtual]

6.183.2.5 virtual Pointer<DataStructure>& activemq::commands::DataResponse::getData ( ) [virtual]

6.183.2.6 virtual unsigned char activemq::commands::DataResponse::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 1783).

6.183.2.7 virtual void activemq::commands::DataResponse::setData (const Pointer< DataStructure > & *data*) [virtual]

6.183.2.8 virtual std::string activemq::commands::DataResponse::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.183.3 Field Documentation

6.183.3.1 Pointer<DataStructure> activemq::commands::DataResponse::data [protected]

6.183.3.2 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**

6.184 activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 865).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/DataResponseMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual ~**DataResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.184.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 865).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.184.2 Constructor & Destructor Documentation

6.184.2.1 activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]

6.184.2.2 virtual activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::∼DataResponseMarshaller ( ) [inline, virtual]

6.184.3 Member Function Documentation

6.184.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1789).

6.184.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.184.3.3 virtual void activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::looseMarshal (OpenWireFormat ∗ *format,* commands::DataStructure ∗ *command,* decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.184.3.4 virtual void activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.184.3.5 virtual int activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.184.3.6 virtual void activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.184.3.7 virtual void activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**DataResponseMarshaller.h**

6.185 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 **tightMarshal1** (**OpenWireFormat** *format, **commands::DataStructure** *command, **utils::BooleanStream** *bs)=0

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** ∗format, **commands::DataStructure** ∗command, **decaf::io::DataOutputStream** ∗ds, **utils::BooleanStream** ∗bs)=0

  Tight Marhsal to the given stream.

- virtual void **tightUnmarshal** (**OpenWireFormat** ∗format, **commands::DataStructure** ∗command, **decaf::io::DataInputStream** ∗dis, **utils::BooleanStream** ∗bs)=0

  Tight Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗format, **commands::DataStructure** ∗command, **decaf::io::DataOutputStream** ∗ds)=0

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗format, **commands::DataStructure** ∗command, **decaf::io::DataInputStream** ∗dis)=0

  Loose Un-marhsal to the given stream.

6.185.1 Detailed Description

Base class for all classes that marshal commands for Openwire.

6.185.2 Constructor & Destructor Documentation

6.185.2.1 virtual activemq::wireformat::openwire::marshal::DataStreamMarshaller::∼DataStreamMarshaller ( ) [inline, virtual]

6.185.3 Member Function Documentation

6.185.3.1 virtual commands::DataStructure∗ activemq::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::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 143), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 221), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 226), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 256), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 291), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 304), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 311), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 325), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 441), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 449), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 733), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 739), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 749), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 757), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 774), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 780), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 790), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 796), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832),

activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller (p. 866), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 944), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1071), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1178), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1214), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1220), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1225), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1231), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1237), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1248), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1302), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1453), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1467), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1473), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1483), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1507), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1531), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1687), **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller** (p. 1695), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1702), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1762), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1768), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1774), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1789), **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller** (p. 1847), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1852), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1887), **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller** (p. 2043), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2153), **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller** (p. 2249), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2282).

6.185.3.2 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::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 143), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 221), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 226), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 256), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 326), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 441), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 449), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 733), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 739), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 749), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 757), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 774), **activemq::wireformat::openwire::marshal::generated::**

::ConsumerIdMarshaller (p. 780), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 790), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 796), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 944), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1178), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1214), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1220), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1226), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1232), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1237), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1248), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1302), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1453), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1467), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1474), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1483), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1507), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1531), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1687), **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller** (p. 1695), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1702), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1762), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1768), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1774), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller** (p. 1847), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1887), **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller** (p. 2043), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2153), **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller** (p. 2249), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2283).

6.185.3.3 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [pure virtual]

Tight Marhsal 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::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 221), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 256), **activemq::wireformat::openwire-**

::marshal::generated::ActiveMQStreamMessageMarshaller (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 326), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 441), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 449), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 734), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 740), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 749), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 757), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 774), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 781), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 790), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 796), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 944), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1178), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1214), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1221), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1226), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1232), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1237), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1248), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1302), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1453), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1467), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1474), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1484), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1507), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1531), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1687), **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller** (p. 1696), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1762), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1769), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1774), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller** (p. 1847), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1887), **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller** (p. 2044), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2153), **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller** (p. 2249), **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2283), **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 297), **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387), **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1487), and **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2146).

6.185.3.4 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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 **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 256), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 326), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 441), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 450), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 734), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 740), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 750), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 757), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 774), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 781), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 791), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 797), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 944), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1214), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1221), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1226), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1232), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1238), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1249), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1302), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1467), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1474), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1484), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1508), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1531), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1687), **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller** (p. 1696), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1762), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1769), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1775), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller** (p. 1847), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1887), **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller** (p. 2044), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2153), **activemq::wireformat::openwire::marshal::generated::-**

WireFormatInfoMarshaller (p. 2250), **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2283), **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298), **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488), and **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2147).

6.185.3.5 virtual int activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [pure virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 257), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 305), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 312), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 326), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 442), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 450), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 734), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 740), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 750), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 758), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 775), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 781), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 791), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 797), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 954), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1215), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1221), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1227), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1232), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1238), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1249), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1303), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1468), **activemq::wireformat::openwire::marshal::generated::-**

MessageDispatchNotificationMarshaller (p. 1475), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1484), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1508), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1532), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1613), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1688), **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller** (p. 1696), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1763), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1769), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1775), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791), **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller** (p. 1848), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1888), **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller** (p. 2044), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2154), **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller** (p. 2250), **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2283), **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 298), **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1488), and **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2147).

6.185.3.6 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2 (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*) [pure virtual]

Tight Marhsal 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::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 145), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 257), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 306), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 313), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 327), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 442), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 450), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 735), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 741), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 750), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 758), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 775), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 782), **activemq::wireformat::openwire::marshal::generated::ConsumerInfo-**

Marshaller (p. 791), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 797), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 954), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1215), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1222), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1227), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1233), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1238), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1249), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1303), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1468), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1475), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1485), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1508), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1532), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1613), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1688), **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller** (p. 1697), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1704), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1763), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1770), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1775), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791), **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller** (p. 1848), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1854), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1888), **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller** (p. 2045), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2154), **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller** (p. 2250), **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2284), **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299), **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390), **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489), and **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2148).

6.185.3.7 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 145),

activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller (p. 223), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 238), **activemq::wireformat::openwire::marshal::generated::ActiveMQQueueMarshaller** (p. 258), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempQueueMarshaller** (p. 306), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempTopicMarshaller** (p. 313), **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::generated::ActiveMQTopicMarshaller** (p. 327), **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller** (p. 442), **activemq::wireformat::openwire::marshal::generated::BrokerInfoMarshaller** (p. 451), **activemq::wireformat::openwire::marshal::generated::ConnectionControlMarshaller** (p. 735), **activemq::wireformat::openwire::marshal::generated::ConnectionErrorMarshaller** (p. 741), **activemq::wireformat::openwire::marshal::generated::ConnectionIdMarshaller** (p. 751), **activemq::wireformat::openwire::marshal::generated::ConnectionInfoMarshaller** (p. 759), **activemq::wireformat::openwire::marshal::generated::ConsumerControlMarshaller** (p. 776), **activemq::wireformat::openwire::marshal::generated::ConsumerIdMarshaller** (p. 782), **activemq::wireformat::openwire::marshal::generated::ConsumerInfoMarshaller** (p. 792), **activemq::wireformat::openwire::marshal::generated::ControlCommandMarshaller** (p. 798), **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 868), **activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller** (p. 945), **activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller** (p. 954), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), **activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1180), **activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller** (p. 1215), **activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller** (p. 1222), **activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller** (p. 1227), **activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller** (p. 1233), **activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller** (p. 1239), **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1250), **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1303), **activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller** (p. 1455), **activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller** (p. 1468), **activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller** (p. 1475), **activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller** (p. 1485), **activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller** (p. 1509), **activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller** (p. 1532), **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1614), **activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller** (p. 1688), **activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller** (p. 1697), **activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller** (p. 1704), **activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller** (p. 1764), **activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller** (p. 1770), **activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller** (p. 1776), **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1792), **activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller** (p. 1848), **activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller** (p. 1854), **activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller** (p. 1889), **activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller** (p. 2045), **activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller** (p. 2154), **activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller** (p. 2251), **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2284), **activemq::wireformat::openwire::marshal::generated::ActiveMQDestinationMarshaller** (p. 202), **activemq::wireformat::openwire::marshal::generated::ActiveMQTempDestinationMarshaller** (p. 299), **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391), **activemq::wireformat::openwire::marshal::generated::MessageMarshaller** (p. 1489), and **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2148).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**DataStreamMarshaller.h**

6.186 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. 877) Type as defined in CommandTypes.h.*
- virtual **DataStructure** * **cloneDataStructure** () **const** =0

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const** =0

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*

6.186.1 Constructor & Destructor Documentation

6.186.1.1 virtual activemq::commands::DataStructure::~DataStructure ( ) [inline, virtual]

6.186.2 Member Function Documentation

6.186.2.1 virtual DataStructure* activemq::commands::DataStructure::cloneDataStructure ( ) const [pure virtual]

Clone this obbject 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::Message** (p. 1417), **activemq::commands::ActiveMQDestination** (p. 193), **activemq::commands::ConsumerInfo** (p. 785), **activemq::commands::MessageId** (p. 1480), **activemq::commands::SessionId** (p. 1844), **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::BrokerInfo** (p. 445), **activemq::commands::ConnectionInfo** (p. 753), **activemq::commands::ProducerId** (p. 1692), **activemq::commands::ConnectionId** (p. 746), **activemq::commands::MessageAck** (p. 1449), **activemq::commands::ConsumerId** (p. 777), **activemq::commands::ConsumerControl** (p. 770), **activemq::commands::JournalTopicAck** (p. 1217), **activemq::commands::ProducerInfo** (p. 1699), **activemq::commands::BrokerError** (p. 434), **activemq::commands::ConnectionControl** (p. 729), **activemq::commands::DestinationInfo** (p. 940), **activemq::commands::MessagePull** (p. 1504), **activemq::commands::SessionInfo** (p. 1850), **activemq::commands::MessageDispatch** (p. 1460), **activemq::commands::MessageDispatchNotification** (p. 1470), **activemq::commands::SubscriptionInfo** (p. 2040), **activemq::commands::XATransactionId** (p. 2278), **activemq::commands::TransactionInfo** (p. 2149), **activemq::commands::ConnectionError** (p. 736), **activemq::commands::JournalQueueAck** (p. 1211), **activemq::commands::JournalTransaction** (p. 1229), **activemq::commands::RemoveSubscriptionInfo** (p. 1765), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::LocalTransactionId** (p. 1299), **activemq::commands::NetworkBridgeFilter** (p. 1528), **activemq::commands::ProducerAck** (p. 1684), **activemq::commands::RemoveInfo** (p. 1759), **activemq::commands::DataArrayResponse** (p. 829), **activemq::commands::DataResponse** (p. 864), **activemq::commands::Discovery-**

Event (p. 950), **activemq::commands::ExceptionResponse** (p. 997), **activemq::commands::PartialCommand** (p. 1609), **activemq::commands::ReplayCommand** (p. 1771), **activemq::commands::BrokerId** (p. 438), **activemq::commands::ControlCommand** (p. 793), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::JournalTrace** (p. 1223), **activemq::commands::Response** (p. 1782), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::FlushCommand** (p. 1069), **activemq::commands::KeepAliveInfo** (p. 1234), **activemq::commands::LastPartialCommand** (p. 1246), **activemq::commands::ShutdownInfo** (p. 1884), **activemq::commands::TransactionId** (p. 2144), **activemq::commands::ActiveMQTempDestination** (p. 294), **activemq::commands::ActiveMQBlobMessage** (p. 123), **activemq::commands::WireFormatInfo** (p. 2242), **activemq::commands::ActiveMQQueue** (p. 250), **activemq::commands::ActiveMQTopic** (p. 322), **activemq::commands::ActiveMQTextMessage** (p. 315), **activemq::commands::ActiveMQMessage** (p. 224), **activemq::commands::ActiveMQTempQueue** (p. 301), **activemq::commands::ActiveMQTempTopic** (p. 308), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::BooleanExpression** (p. 427).

6.186.2.2 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::Message** (p. 1417), **activemq::commands::ActiveMQDestination** (p. 193), **activemq::commands::ConsumerInfo** (p. 785), **activemq::commands::BrokerError** (p. 434), **activemq::commands::MessageId** (p. 1481), **activemq::commands::SessionId** (p. 1844), **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::BrokerInfo** (p. 445), **activemq::commands::ConnectionInfo** (p. 753), **activemq::commands::ProducerId** (p. 1693), **activemq::commands::ConnectionId** (p. 747), **activemq::commands::MessageAck** (p. 1449), **activemq::commands::ConsumerId** (p. 778), **activemq::commands::ConsumerControl** (p. 771), **activemq::commands::JournalTopicAck** (p. 1217), **activemq::commands::ProducerInfo** (p. 1699), **activemq::commands::ConnectionControl** (p. 730), **activemq::commands::DestinationInfo** (p. 940), **activemq::commands::MessagePull** (p. 1504), **activemq::commands::SessionInfo** (p. 1850), **activemq::commands::MessageDispatch** (p. 1460), **activemq::commands::MessageDispatchNotification** (p. 1470), **activemq::commands::SubscriptionInfo** (p. 2040), **activemq::commands::XATransactionId** (p. 2279), **activemq::commands::TransactionInfo** (p. 2150), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::JournalQueueAck** (p. 1211), **activemq::commands::JournalTransaction** (p. 1229), **activemq::commands::RemoveSubscriptionInfo** (p. 1765), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::LocalTransactionId** (p. 1299), **activemq::commands::NetworkBridgeFilter** (p. 1528), **activemq::commands::ProducerAck** (p. 1684), **activemq::commands::RemoveInfo** (p. 1759), **activemq::commands::DataArrayResponse** (p. 829), **activemq::commands::DataResponse** (p. 864), **activemq::commands::DiscoveryEvent** (p. 951), **activemq::commands::ExceptionResponse** (p. 997), **activemq::commands::PartialCommand** (p. 1609), **activemq::commands::ReplayCommand** (p. 1771), **activemq::commands::ActiveMQTempDestination** (p. 295), **activemq::commands::BrokerId** (p. 439), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::JournalTrace** (p. 1223), **activemq::commands::Response** (p. 1782), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::FlushCommand** (p. 1069), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::LastPartialCommand** (p. 1246), **activemq::commands::ShutdownInfo** (p. 1884), **activemq::commands::TransactionId** (p. 2144), **activemq::commands::ActiveMQBlobMessage** (p. 123), **activemq::commands::WireFormatInfo** (p. 2242), **activemq::commands::ActiveMQQueue** (p. 250), **activemq::commands::ActiveMQTopic** (p. 322), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQTempQueue** (p. 301), **activemq::commands::ActiveMQTempTopic** (p. 308), **activemq::commands::ActiveMQObjectMessage** (p. 234), **activemq::commands::BaseCommand** (p. 382), and **activemq::commands::ActiveMQMessage** (p. 224).

6.186.2.3 virtual bool activemq::commands::DataStructure::equals (const DataStructure * *value*) const [pure virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implemented in **activemq::commands::Message** (p. 1417), **activemq::commands::ActiveMQDestination** (p. 194), **activemq::commands::ConsumerInfo** (p. 785), **activemq::commands::MessageId** (p. 1481), **activemq::commands::SessionId** (p. 1844), **activemq::commands::BaseCommand** (p. 382), **activemq::commands::ActiveMQBytesMessage** (p. 132), **activemq::commands::BrokerInfo** (p. 445), **activemq::commands::ConnectionInfo** (p. 753), **activemq::commands::ProducerId** (p. 1693), **activemq::commands::ConnectionId** (p. 747), **activemq::commands::MessageAck** (p. 1450), **activemq::commands::ConsumerId** (p. 778), **activemq::commands::ConsumerControl** (p. 771), **activemq::commands::JournalTopicAck** (p. 1217), **activemq::commands::ProducerInfo** (p. 1699), **activemq::commands::ConnectionControl** (p. 730), **activemq::commands::DestinationInfo** (p. 941), **activemq::commands::MessagePull** (p. 1504), **activemq::commands::SessionInfo** (p. 1850), **activemq::commands::MessageDispatch** (p. 1460), **activemq::commands::MessageDispatchNotification** (p. 1470), **activemq::commands::SubscriptionInfo** (p. 2041), **activemq::commands::XATransactionId** (p. 2279), **activemq::commands::TransactionInfo** (p. 2150), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::JournalQueueAck** (p. 1211), **activemq::commands::JournalTransaction** (p. 1229), **activemq::commands::RemoveSubscriptionInfo** (p. 1765), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ActiveMQTempDestination** (p. 295), **activemq::commands::LocalTransactionId** (p. 1299), **activemq::commands::NetworkBridgeFilter** (p. 1528), **activemq::commands::ProducerAck** (p. 1684), **activemq::commands::RemoveInfo** (p. 1759), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 864), **activemq::commands::DiscoveryEvent** (p. 951), **activemq::commands::ExceptionResponse** (p. 998), **activemq::commands::PartialCommand** (p. 1609), **activemq::commands::ReplayCommand** (p. 1772), **activemq::commands::BrokerId** (p. 439), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::JournalTrace** (p. 1224), **activemq::commands::Response** (p. 1783), **activemq::commands::FlushCommand** (p. 1069), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::LastPartialCommand** (p. 1246), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::TransactionId** (p. 2144), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230), **activemq::commands::ActiveMQBlobMessage** (p. 124), **activemq::commands::WireFormatInfo** (p. 2242), **activemq::commands::ActiveMQQueue** (p. 250), **activemq::commands::ActiveMQTopic** (p. 323), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQTempQueue** (p. 301), **activemq::commands::ActiveMQTempTopic** (p. 309), **activemq::commands::ActiveMQObjectMessage** (p. 234), **activemq::commands::ActiveMQMessage** (p. 224), and **activemq::commands::BooleanExpression** (p. 427).

6.186.2.4 virtual unsigned char activemq::commands::DataStructure::getDataStructureType ( ) const [pure virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implemented in **activemq::commands::Message** (p. 1419), **activemq::commands::ActiveMQDestination** (p. 195), **activemq::commands::ConsumerInfo** (p. 786), **activemq::commands::MessageId** (p. 1481), **activemq::commands::SessionId** (p. 1845), **activemq::commands::ActiveMQBytesMessage** (p. 132), **activemq::commands::BrokerInfo** (p. 446), **activemq::commands::ConnectionInfo** (p. 753), **activemq::commands::ProducerId** (p. 1693), **activemq::commands::ConnectionId** (p. 747), **activemq::commands::MessageAck** (p. 1450), **activemq::commands::ConsumerId** (p. 778), **activemq::commands::ConsumerControl** (p. 771), **activemq::commands::JournalTopicAck** (p. 1218), **activemq::commands::ProducerInfo** (p. 1699), **activemq::commands::ConnectionControl** (p. 730), **activemq::commands::DestinationInfo** (p. 941), **activemq::commands::MessagePull** (p. 1504), **activemq::commands::SessionInfo** (p. 1850), **activemq::commands::MessageDispatch** (p. 1461), **activemq::commands::MessageDispatchNotification** (p. 1471), **activemq::commands::SubscriptionInfo** (p. 2041), **activemq::commands::XATransactionId** (p. 2280), **activemq::commands::TransactionInfo** (p. 2150), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::JournalQueueAck** (p. 1212), **activemq::commands::JournalTransaction** (p. 1229), **activemq::commands::RemoveSubscriptionInfo** (p. 1766), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::LocalTransactionId** (p. 1300), **activemq::commands::NetworkBridgeFilter** (p. 1529), **activemq::commands::ProducerAck** (p. 1684), **activemq::commands::RemoveInfo** (p. 1760), **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 864), **activemq::commands::DiscoveryEvent** (p. 951), **activemq::commands::ExceptionResponse** (p. 998), **activemq::commands::PartialCommand** (p. 1610), **activemq::commands::ReplayCommand** (p. 1772), **activemq::commands::BrokerError** (p. 434), **activemq::commands::BrokerId** (p. 439), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::JournalTrace** (p. 1224), **activemq::commands::Response** (p. 1783), **activemq::commands::FlushCommand** (p. 1070), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::LastPartialCommand** (p. 1246), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::TransactionId** (p. 2145), **activemq::commands::ActiveMQTempDestination** (p. 295), **activemq::commands::ActiveMQMapMessage** (p. 213), **activemq::commands::ActiveMQBlobMessage** (p. 124), **activemq::commands::WireFormatInfo** (p. 2243), **activemq::commands::ActiveMQQueue** (p. 251), **activemq::commands::ActiveMQTopic** (p. 323), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQTempQueue** (p. 302), **activemq::commands::ActiveMQTempTopic** (p. 309), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 225).

6.186.2.5 virtual std::string activemq::commands::DataStructure::toString ( ) const [pure virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implemented in **activemq::commands::Message** (p. 1424), **activemq::commands::ActiveMQDestination** (p. 198), **activemq::commands::ConsumerInfo** (p. 788), **activemq::commands::MessageId** (p. 1482), **activemq::commands::SessionId** (p. 1845), **activemq::commands::ActiveMQBytesMessage** (p. 138), **activemq::commands::BrokerInfo** (p. 447), **activemq::commands::ConnectionInfo** (p. 755), **activemq::commands::ProducerId** (p. 1694), **activemq::commands::ConnectionId** (p. 747), **activemq::commands::MessageAck** (p. 1451), **activemq::commands::ConsumerId** (p. 779), **activemq::commands::ConsumerControl** (p. 772), **activemq::commands::JournalTopicAck** (p. 1219), **activemq::commands::ProducerInfo** (p. 1700), **activemq::commands::ConnectionControl** (p. 731), **activemq::commands::DestinationInfo** (p. 942), **activemq::commands::MessagePull** (p. 1505), **activemq::commands::SessionInfo** (p. 1851), **activemq::commands::MessageDispatch** (p. 1461), **activemq::commands::MessageDispatchNotification** (p. 1472), **activemq::commands::SubscriptionInfo** (p. 2042), **activemq::commands::XATransactionId** (p. 2281), **activemq::commands::TransactionInfo** (p. 2151), **activemq::commands::ConnectionError** (p. 737), **activemq::commands::JournalQueueAck** (p. 1212), **activemq::commands::JournalTransaction** (p. 1230), **activemq::commands::RemoveSubscriptionInfo** (p. 1766), **activemq::commands::ActiveMQStreamMessage** (p. 286), **activemq::commands::ActiveMQTempDestination** (p. 296), **activemq::commands-**

::LocalTransactionId (p. 1300), **activemq::commands::NetworkBridgeFilter** (p. 1529), **activemq::commands::ProducerAck** (p. 1685), **activemq::commands::RemoveInfo** (p. 1760), **activemq::commands::ActiveMQMapMessage** (p. 219), **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 865), **activemq::commands::DiscoveryEvent** (p. 951), **activemq::commands::ExceptionResponse** (p. 998), **activemq::commands::PartialCommand** (p. 1610), **activemq::commands::ReplayCommand** (p. 1772), **activemq::commands::BrokerId** (p. 439), **activemq::commands::ControlCommand** (p. 794), **activemq::commands::IntegerResponse** (p. 1176), **activemq::commands::JournalTrace** (p. 1224), **activemq::commands::Response** (p. 1783), **activemq::commands::FlushCommand** (p. 1070), **activemq::commands::KeepAliveInfo** (p. 1235), **activemq::commands::LastPartialCommand** (p. 1247), **activemq::commands::ShutdownInfo** (p. 1885), **activemq::commands::TransactionId** (p. 2145), **activemq::commands::BaseCommand** (p. 386), **activemq::commands::ActiveMQBlobMessage** (p. 125), **activemq::commands::Command** (p. 674), **activemq::commands::WireFormatInfo** (p. 2247), **activemq::commands::ActiveMQQueue** (p. 252), **activemq::commands::BaseDataStructure** (p. 411), **activemq::commands::ActiveMQTopic** (p. 324), **activemq::commands::ActiveMQTextMessage** (p. 317), **activemq::commands::ActiveMQTempQueue** (p. 303), **activemq::commands::ActiveMQTempTopic** (p. 310), **activemq::commands::ActiveMQObjectMessage** (p. 235), **activemq::commands::ActiveMQMessage** (p. 225), and **activemq::commands::BooleanExpression** (p. 428).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataStructure.h**

6.187 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. 882) 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. 882) object to a String of the form:*
- virtual int **compareTo** (**const Date** &value) **const**

- virtual bool **equals** (**const Date** &value) **const**
- virtual bool **operator==** (**const Date** &value) **const**
- virtual bool **operator<** (**const Date** &value) **const**

6.187.1 Detailed Description

Wrapper class around a time value in milliseconds.

This class is comparable to Java's java.util.Date class.

Since

1.0

6.187.2 Constructor & Destructor Documentation

6.187.2.1 decaf::util::Date::Date ( )

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

6.187.2.2 decaf::util::Date::Date (long long *milliseconds*)

Constructs the date with a given time value.

Parameters

milliseconds	The time in milliseconds;

6.187.2.3 decaf::util::Date::Date (const Date & *source*)

Copy constructor.

Parameters

source	The **Date** (p. 882) instance to copy into this one.

6.187.2.4 virtual decaf::util::Date::~Date ( ) [virtual]

6.187.3 Member Function Documentation

6.187.3.1 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.

6.187.3.2 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.

6.187.3.3 virtual int decaf::util::Date::compareTo (const Date & *value*) const [virtual]

6.187.3.4 virtual bool decaf::util::Date::equals (const Date & *value*) const [virtual]

6.187.3.5 long long decaf::util::Date::getTime ( ) const

Gets the underlying time.

Returns

The underlying time value in milliseconds.

6.187.3.6 virtual bool decaf::util::Date::operator< (const Date & *value*) const [virtual]

6.187.3.7 Date& decaf::util::Date::operator= (const Date & *value*)

Assigns the value of one **Date** (p. 882) object to another.

Parameters

value	The value to be copied into this **Date** (p. 882) object.

Returns

reference to this object with the newly assigned value.

6.187.3.8 virtual bool decaf::util::Date::operator== (const Date & *value*) const [virtual]

6.187.3.9 void decaf::util::Date::setTime (long long *milliseconds*)

Sets the underlying time.

Parameters

milliseconds	The underlying time value in milliseconds.

6.187.3.10 std::string decaf::util::Date::toString ( ) const

Converts this **Date** (p. 882) 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. 882) object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

6.188 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.
- **decaf::util::concurrent::Mutex** * **getGlobalLock** ()

  Gets a pointer to the Decaf Runtime's Global Lock object, this can be used by Decaf APIs to synchronize around certain actions such as adding or acquiring a resource that must be Thread safe.

6.188.1 Detailed Description

Handles APR initialization and termination.

6.188.2 Constructor & Destructor Documentation

6.188.2.1 decaf::internal::DecafRuntime::DecafRuntime ( )

Initializes the APR Runtime for a library.

6.188.2.2 virtual decaf::internal::DecafRuntime::~DecafRuntime ( ) [virtual]

Terminates the APR Runtime for a library.

6.188.3 Member Function Documentation

6.188.3.1 decaf::util::concurrent::Mutex* decaf::internal::DecafRuntime::getGlobalLock ( )

Gets a pointer to the Decaf Runtime's Global Lock object, this can be used by Decaf APIs to synchronize around certain actions such as adding or acquiring a resource that must be Thread safe.

The pointer returned is owned by the Decaf runtime and should not be deleted or copied by the caller.

Returns

a pointer to the Decaf Runtime's global Lock instance.

6.188.3.2 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**

6.189 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** (p. 2074)'s thread has exited.*
- virtual void **wakeup** ()

  *Signal the **TaskRunner** (p. 2074) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2073) 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.

6.189.1 Constructor & Destructor Documentation

6.189.1.1 activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner (Task * *task*)

6.189.1.2 virtual activemq::threads::DedicatedTaskRunner::~DedicatedTaskRunner ( ) [virtual]

6.189.2 Member Function Documentation

6.189.2.1 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. 1793).

6.189.2.2 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.

Implements **activemq::threads::TaskRunner** (p. 2074).

6.189.2.3 virtual void activemq::threads::DedicatedTaskRunner::shutdown ( ) [virtual]

Shutdown once the task has finished and the **TaskRunner** (p. 2074)'s thread has exited.

Implements **activemq::threads::TaskRunner** (p. 2074).

6.189.2.4 virtual void activemq::threads::DedicatedTaskRunner::wakeup ( ) [virtual]

Signal the **TaskRunner** (p. 2074) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2073) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 2074).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**DedicatedTaskRunner.h**

6.190 activemq::core::policies::DefaultPrefetchPolicy Class Reference

`#include <src/main/activemq/core/policies/DefaultPrefetchPolicy.h>`

Inheritance diagram for activemq::core::policies::DefaultPrefetchPolicy:

Public Member Functions

- **DefaultPrefetchPolicy** ()
- virtual ~**DefaultPrefetchPolicy** ()
- virtual void **setDurableTopicPrefetch** (int value)

Sets the amount of prefetched messages for a Durable Topic.

- virtual int **getDurableTopicPrefetch** () **const**

  Gets the amount of messages to prefetch for a Durable Topic.

- virtual void **setQueuePrefetch** (int value)

  Sets the amount of prefetched messages for a Queue.

- virtual int **getQueuePrefetch** () **const**

  Gets the amount of messages to prefetch for a Queue.

- virtual void **setQueueBrowserPrefetch** (int value)

  Sets the amount of prefetched messages for a Queue Browser.

- virtual int **getQueueBrowserPrefetch** () **const**

  Gets the amount of messages to prefetch for a Queue Browser.

- virtual void **setTopicPrefetch** (int value)

  Sets the amount of prefetched messages for a Topic.

- virtual int **getTopicPrefetch** () **const**

  Gets the amount of messages to prefetch for a Topic.

- virtual int **getMaxPrefetchLimit** (int value) **const**

  Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

- virtual **PrefetchPolicy** * **clone** () **const**

  Clone the Policy and return a new pointer to that clone.

Static Public Attributes

- static int **MAX_PREFETCH_SIZE**
- static int **DEFAULT_DURABLE_TOPIC_PREFETCH**
- static int **DEFAULT_QUEUE_PREFETCH**
- static int **DEFAULT_QUEUE_BROWSER_PREFETCH**
- static int **DEFAULT_TOPIC_PREFETCH**

6.190.1 Constructor & Destructor Documentation

6.190.1.1 activemq::core::policies::DefaultPrefetchPolicy::DefaultPrefetchPolicy ( )

6.190.1.2 virtual activemq::core::policies::DefaultPrefetchPolicy::~DefaultPrefetchPolicy ( ) [virtual]

6.190.2 Member Function Documentation

6.190.2.1 virtual PrefetchPolicy* activemq::core::policies::DefaultPrefetchPolicy::clone ( ) const [virtual]

Clone the Policy and return a new pointer to that clone.

Returns

pointer to a new **PrefetchPolicy** (p. 1635) instance that is a clone of this one.

Implements **activemq::core::PrefetchPolicy** (p. 1636).

6.190.2.2 virtual int activemq::core::policies::DefaultPrefetchPolicy::getDurableTopicPrefetch ( ) const [inline, virtual]

Gets the amount of messages to prefetch for a Durable Topic.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1636).

6.190.2.3 virtual int activemq::core::policies::DefaultPrefetchPolicy::getMaxPrefetchLimit (int *value*) const [inline, virtual]

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

Returns

the allowable value for a prefetch limit, either requested or the max.

Implements **activemq::core::PrefetchPolicy** (p. 1636).

6.190.2.4 virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueueBrowserPrefetch ( ) const [inline, virtual]

Gets the amount of messages to prefetch for a Queue Browser.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1637).

6.190.2.5 virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueuePrefetch ( ) const [inline, virtual]

Gets the amount of messages to prefetch for a Queue.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1637).

6.190.2.6 virtual int activemq::core::policies::DefaultPrefetchPolicy::getTopicPrefetch ( ) const [inline, virtual]

Gets the amount of messages to prefetch for a Topic.

Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1637).

6.190.2.7 virtual void activemq::core::policies::DefaultPrefetchPolicy::setDurableTopicPrefetch (int *value*) [inline, virtual]

Sets the amount of prefetched messages for a Durable Topic.

Parameters

value	The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1637).

6.190.2.8 virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueueBrowserPrefetch (int *value*) [inline, virtual]

Sets the amount of prefetched messages for a Queue Browser.

Parameters

value	The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1637).

6.190.2.9 virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueuePrefetch (int *value*) [inline, virtual]

Sets the amount of prefetched messages for a Queue.

Parameters

value	The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1638).

6.190.2.10 virtual void activemq::core::policies::DefaultPrefetchPolicy::setTopicPrefetch (int *value*) [inline, virtual]

Sets the amount of prefetched messages for a Topic.

Parameters

value	The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 1638).

6.190.3 Field Documentation

6.190.3.1 int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_DURABLE_TOPIC_PREFETCH [static]

6.190.3.2 int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_BROWSER_PREFETCH [static]

6.190.3.3 int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_PREFETCH [static]

6.190.3.4 int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_TOPIC_PREFETCH [static]

6.190.3.5 int activemq::core::policies::DefaultPrefetchPolicy::MAX_PREFETCH_SIZE [static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/policies/**DefaultPrefetchPolicy.h**

6.191 activemq::core::policies::DefaultRedeliveryPolicy Class Reference

`#include <src/main/activemq/core/policies/DefaultRedeliveryPolicy.h>`

Inheritance diagram for activemq::core::policies::DefaultRedeliveryPolicy:

Public Member Functions

- **DefaultRedeliveryPolicy** ()
- virtual ~**DefaultRedeliveryPolicy** ()
- virtual double **getBackOffMultiplier** () **const**
- virtual void **setBackOffMultiplier** (double value)

  Sets the Back-Off Multiplier for Message Redelivery.
- virtual short **getCollisionAvoidancePercent** () **const**
- virtual void **setCollisionAvoidancePercent** (short value)
- virtual long long **getInitialRedeliveryDelay** () **const**

  Gets the initial time that redelivery of messages is delayed.
- virtual void **setInitialRedeliveryDelay** (long long value)

  Sets the initial time that redelivery will be delayed.
- virtual long long **getRedeliveryDelay** () **const**

  Gets the time that redelivery of messages is delayed.
- virtual void **setRedeliveryDelay** (long long value)

  Sets the time that redelivery will be delayed.
- virtual int **getMaximumRedeliveries** () **const**

  Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.
- virtual void **setMaximumRedeliveries** (int value)

  Sets the Maximum allowable redeliveries for a Message.
- virtual bool **isUseCollisionAvoidance** () **const**
- virtual void **setUseCollisionAvoidance** (bool value)
- virtual bool **isUseExponentialBackOff** () **const**
- virtual void **setUseExponentialBackOff** (bool value)
- virtual long long **getNextRedeliveryDelay** (long long previousDelay)

  Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.
- virtual **RedeliveryPolicy** * **clone** () **const**

  Create a copy of this Policy and return it.

6.191.1 Constructor & Destructor Documentation

6.191.1.1 activemq::core::policies::DefaultRedeliveryPolicy::DefaultRedeliveryPolicy ( )

6.191.1.2 virtual activemq::core::policies::DefaultRedeliveryPolicy::~DefaultRedeliveryPolicy ( ) [virtual]

6.191.2 Member Function Documentation

6.191.2.1 virtual RedeliveryPolicy* activemq::core::policies::DefaultRedeliveryPolicy::clone ( ) const [virtual]

Create a copy of this Policy and return it.

Returns

pointer to a new **RedeliveryPolicy** (p. 1744) that is a copy of this one.

Implements **activemq::core::RedeliveryPolicy** (p. 1745).

6.191.2.2 virtual double activemq::core::policies::DefaultRedeliveryPolicy::getBackOffMultiplier ( ) const [inline, virtual]

Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 1746).

6.191.2.3 virtual short activemq::core::policies::DefaultRedeliveryPolicy::getCollisionAvoidancePercent ( ) const [virtual]

Returns

the currently set Collision Avoidance percentage.

Implements **activemq::core::RedeliveryPolicy** (p. 1746).

6.191.2.4 virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getInitialRedeliveryDelay ( ) const [inline, virtual]

Gets the initial time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed initially.

Implements **activemq::core::RedeliveryPolicy** (p. 1746).

6.191.2.5 virtual int activemq::core::policies::DefaultRedeliveryPolicy::getMaximumRedeliveries ( ) const [inline, virtual]

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

Returns

maximum allowed redeliveries for a message.

Implements **activemq::core::RedeliveryPolicy** (p. 1746).

6.191.2.6 virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getNextRedeliveryDelay (long long *previousDelay*) [virtual]

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

Parameters

previousDelay	The last delay that was used between message redeliveries.

Returns

the new delay to use before attempting another redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 1746).

6.191.2.7 virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getRedeliveryDelay ( ) const [inline, virtual]

Gets the time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed.

Implements **activemq::core::RedeliveryPolicy** (p. 1747).

6.191.2.8 virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseCollisionAvoidance ( ) const [inline, virtual]

Returns

whether or not collision avoidance is enabled for this Policy.

Implements **activemq::core::RedeliveryPolicy** (p. 1747).

6.191.2.9 virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseExponentialBackOff ( ) const [inline, virtual]

Returns

whether or not the exponential back off option is enabled.

Implements **activemq::core::RedeliveryPolicy** (p. 1747).

6.191.2.10 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setBackOffMultiplier (double *value*) [inline, virtual]

Sets the Back-Off Multiplier for Message Redelivery.

Parameters

value	The new value for the back-off multiplier.

Implements **activemq::core::RedeliveryPolicy** (p. 1747).

6.191.2.11 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setCollisionAvoidancePercent (short *value*) [virtual]

Parameters

value	The collision avoidance percentage setting.

Implements **activemq::core::RedeliveryPolicy** (p. 1747).

6.191.2.12 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setInitialRedeliveryDelay (long long *value*) [inline, virtual]

Sets the initial time that redelivery will be delayed.

Parameters

value	Time in Milliseconds to wait before starting redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 1748).

6.191.2.13 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setMaximumRedeliveries (int *maximumRedeliveries*) [inline, virtual]

Sets the Maximum allowable redeliveries for a Message.

Parameters

maximum-Redeliveries	The maximum number of times that a message will be redelivered.

Implements **activemq::core::RedeliveryPolicy** (p. 1748).

6.191.2.14 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setRedeliveryDelay (long long *value*) [inline, virtual]

Sets the time that redelivery will be delayed.

Parameters

value	Time in Milliseconds to wait before the next redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 1748).

6.191.2.15 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseCollisionAvoidance (bool *value*) [inline, virtual]

Parameters

value	Enable or Disable collision avoidance for this Policy.

Implements **activemq::core::RedeliveryPolicy** (p. 1748).

6.191.2.16 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseExponentialBackOff (bool *value*) [inline, virtual]

Parameters

value	Enable or Disable the exponential back off multiplier option.

Implements **activemq::core::RedeliveryPolicy** (p. 1748).

The documentation for this class was generated from the following file:

- src/main/activemq/core/policies/**DefaultRedeliveryPolicy.h**

6.192 decaf::internal::net::DefaultServerSocketFactory Class Reference

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

`#include <src/main/decaf/internal/net/DefaultServerSocketFactory.h>`

Inheritance diagram for decaf::internal::net::DefaultServerSocketFactory:

Public Member Functions

- **DefaultServerSocketFactory** ()
- virtual ∼**DefaultServerSocketFactory** ()
- virtual **decaf::net::ServerSocket** ∗ **createServerSocket** ()

  *Create a new **ServerSocket** (p. 1816) that is unbound.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** ∗ **createServerSocket** (int port)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** ∗ **createServerSocket** (int port, int backlog)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will use the specified connection backlog setting.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*
backlog	*The number of pending connect request the **ServerSocket** (p. 1816) can queue.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** * **createServerSocket** (int port, int backlog, **const decaf::net::InetAddress** *address)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 1816) will listen on all interfaces.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*
backlog	*The number of pending connect request the **ServerSocket** (p. 1816) can queue.*
address	*The address of the interface on the local machine to bind to.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

6.192.1 Detailed Description

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

Since

1.0

6.192.2 Constructor & Destructor Documentation

6.192.2.1 decaf::internal::net::DefaultServerSocketFactory::DefaultServerSocketFactory ( )

6.192.2.2 virtual decaf::internal::net::DefaultServerSocketFactory::~DefaultServerSocketFactory ( ) [virtual]

6.192.3 Member Function Documentation

6.192.3.1 virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( ) [virtual]

Create a new **ServerSocket** (p. 1816) that is unbound.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 1824).

6.192.3.2 virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int *port*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1824).

6.192.3.3 virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int *port,* int *backlog*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will use the specified connection backlog setting.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1825).

6.192.3.4 virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int *port,* int *backlog,* const decaf::net::InetAddress * *address*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 1816) will listen on all interfaces.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.

address	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1825).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**DefaultServerSocketFactory.h**

6.193 decaf::internal::net::DefaultSocketFactory Class Reference

SocketFactory implementation that is used to create Sockets.

`#include <src/main/decaf/internal/net/DefaultSocketFactory.h>`

Inheritance diagram for decaf::internal::net::DefaultSocketFactory:

Public Member Functions

- **DefaultSocketFactory** ()
- virtual ∼**DefaultSocketFactory** ()
- virtual **decaf::net::Socket** ∗ **createSocket** ()

  *Creates an unconnected **Socket** (p. 1900) object.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if the **Socket** (p. 1900) cannot be created.*

- virtual **decaf::net::Socket** ∗ **createSocket** (**const decaf::net::InetAddress** ∗host, int port)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host to connect the socket to.*
port	*The port on the remote host to connect to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** ∗ **createSocket** (**const decaf::net::InetAddress** ∗host, int port, **const decaf::net::InetAddress** ∗ifAddress, int localPort)

*Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*
*The **Socket** (p. 1900) will be bound to the specified local address and port.*

Parameters

host	*The host to connect the socket to.*
port	*The port on the remote host to connect to.*
ifAddress	*The address on the local machine to bind the **Socket** (p. 1900) to.*
localPort	*The local port to bind the **Socket** (p. 1900) to.*

Returns

*a new **Socket** (p. 1900) object, caller must free this object when done.*

Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** * **createSocket** (**const** std::string &name, int port)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host name or IP address to connect the socket to.*
port	*The port on the remote host to connect to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** * **createSocket** (**const** std::string &name, int port, **const decaf::net::InetAddress** *ifAddress, int localPort)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host name or IP address to connect the socket to.*
port	*The port on the remote host to connect to.*
ifAddress	*The address on the local machine to bind the **Socket** (p. 1900) to.*
localPort	*The local port to bind the **Socket** (p. 1900) to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

6.193.1 Detailed Description

SocketFactory implementation that is used to create Sockets.

Since

1.0

6.193.2 Constructor & Destructor Documentation

6.193.2.1 decaf::internal::net::DefaultSocketFactory::DefaultSocketFactory ( )

6.193.2.2 virtual decaf::internal::net::DefaultSocketFactory::~DefaultSocketFactory ( ) [virtual]

6.193.3 Member Function Documentation

6.193.3.1 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket ( ) [virtual]

Creates an unconnected **Socket** (p. 1900) object.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if the **Socket** (p. 1900) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 1917).

6.193.3.2 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const decaf::net::InetAddress * *host,* int *port*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1918).

6.193.3.3 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const decaf::net::InetAddress * *host,* int *port,* const decaf::net::InetAddress * *ifAddress,* int *localPort*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

The **Socket** (p. 1900) will be bound to the specified local address and port.

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1918).

6.193.3.4 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & *name,* int *port*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1919).

6.193.3.5 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & *name,* int *port,* const decaf::net::InetAddress * *ifAddress,* int *localPort*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1919).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**DefaultSocketFactory.h**

6.194 decaf::internal::net::ssl::DefaultSSLContext Class Reference

Default SSLContext manager for the Decaf library.

`#include <src/main/decaf/internal/net/ssl/DefaultSSLContext.h>`

Public Member Functions

- virtual ~**DefaultSSLContext** ()

Static Public Member Functions

- static **decaf::net::ssl::SSLContext** * **getContext** ()

Protected Member Functions

- **DefaultSSLContext** ()

6.194.1 Detailed Description

Default SSLContext manager for the Decaf library.

If the user doesn't supply or specify the SSLContext that they wish to use then we load the Decaf library's default SSLContext using whatever SSL provider is enabled an preferred.

Since

1.0

6.194.2 Constructor & Destructor Documentation

6.194.2.1 decaf::internal::net::ssl::DefaultSSLContext::DefaultSSLContext ( ) [protected]

6.194.2.2 virtual decaf::internal::net::ssl::DefaultSSLContext::~DefaultSSLContext ( ) [virtual]

6.194.3 Member Function Documentation

6.194.3.1 static decaf::net::ssl::SSLContext* decaf::internal::net::ssl::DefaultSSLContext::getContext ( ) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/**DefaultSSLContext.h**

6.195 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

`#include <src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h>`

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLServerSocketFactory:

Public Member Functions

- **DefaultSSLServerSocketFactory** (**const** std::string &errorMessage)
- virtual ~**DefaultSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket** * **createServerSocket** ()

  *Create a new **ServerSocket** (p. 1816) that is unbound.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** * **createServerSocket** (int port)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** * **createServerSocket** (int port, int backlog)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will use the specified connection backlog setting.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*
backlog	*The number of pending connect request the **ServerSocket** (p. 1816) can queue.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** * **createServerSocket** (int port, int backlog, **const decaf::net::InetAddress** *address)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 1816) will listen on all interfaces.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*
backlog	*The number of pending connect request the **ServerSocket** (p. 1816) can queue.*
address	*The address of the interface on the local machine to bind to.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

  Returns the list of cipher suites which are enabled by default.
  Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

  Returns

  an STL vector containing the list of cipher suites enabled by default.

  See also

  ***getSupportedCipherSuites()** (p. 1947)*

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

  Returns the names of the cipher suites which could be enabled for use on an SSL connection.
  Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

  Returns

  an STL vector containing the list of supported cipher suites.

  See also

  ***getDefaultCipherSuites()** (p. 1947)*

6.195.1 Detailed Description

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Since

1.0

6.195.2 Constructor & Destructor Documentation

6.195.2.1 decaf::internal::net::ssl::DefaultSSLServerSocketFactory::DefaultSSLServerSocketFactory (const std::string & *errorMessage*)

6.195.2.2 virtual decaf::internal::net::ssl::DefaultSSLServerSocketFactory::~DefaultSSLServerSocketFactory ( ) [virtual]

6.195.3 Member Function Documentation

6.195.3.1 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( ) [virtual]

Create a new **ServerSocket** (p. 1816) that is unbound.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 1824).

6.195.3.2 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int *port*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1824).

6.195.3.3 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int *port,* int *backlog*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will use the specified connection backlog setting.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1825).

6.195.3.4 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket (int *port,* int *backlog,* const decaf::net::InetAddress * *address*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 1816) will listen on all interfaces.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.
address	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1825).

6.195.3.5 virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getDefaultCipherSuites ( ) [virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 1947)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 1947).

6.195.3.6 virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getSupportedCipherSuites ( ) [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 1947)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 1947).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/**DefaultSSLServerSocketFactory.h**

6.196 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

`#include <src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h>`

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLSocketFactory:

Public Member Functions

- **DefaultSSLSocketFactory** (**const** std::string &errorMessage)
- virtual ∼**DefaultSSLSocketFactory** ()
- virtual **decaf::net::Socket** ∗ **createSocket** ()

  *Creates an unconnected **Socket** (p. 1900) object.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if the **Socket** (p. 1900) cannot be created.*

- virtual **decaf::net::Socket** ∗ **createSocket** (**const decaf::net::InetAddress** ∗host, int port)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host to connect the socket to.*
port	*The port on the remote host to connect to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** ∗ **createSocket** (**const decaf::net::InetAddress** ∗host, int port, **const decaf::net::InetAddress** ∗ifAddress, int localPort)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*
  *The **Socket** (p. 1900) will be bound to the specified local address and port.*

Parameters

host	*The host to connect the socket to.*
port	*The port on the remote host to connect to.*
ifAddress	*The address on the local machine to bind the **Socket** (p. 1900) to.*
localPort	*The local port to bind the **Socket** (p. 1900) to.*

Returns

*a new **Socket** (p. 1900) object, caller must free this object when done.*

Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** * **createSocket** (**const** std::string &name, int port)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host name or IP address to connect the socket to.*
port	*The port on the remote host to connect to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** * **createSocket** (**const** std::string &name, int port, **const decaf::net::InetAddress** *ifAddress, int localPort)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host name or IP address to connect the socket to.*
port	*The port on the remote host to connect to.*
ifAddress	*The address on the local machine to bind the **Socket** (p. 1900) to.*
localPort	*The local port to bind the **Socket** (p. 1900) to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

  Returns the list of cipher suites which are enabled by default.
  Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

  Returns

  an STL vector containing the list of cipher suites enabled by default.

  See also

  ***getSupportedCipherSuites()** (p. 1956)*

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

  Returns the names of the cipher suites which could be enabled for use on an SSL connection.
  Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() *(p. 1956)*

- virtual **decaf::net::Socket** * **createSocket** (**decaf::net::Socket** *socket, std::string host, int port, bool autoClose)

  Returns a socket layered over an existing socket connected to the named host, at the given port.
  This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

  Parameters

socket	*The existing socket to layer over.*
host	*The server host the original **Socket** (p. 1900) is connected to.*
port	*The server port the original **Socket** (p. 1900) is connected to.*
autoClose	*Should the layered over **Socket** (p. 1900) be closed when the topmost socket is closed.*

  Returns

  *a new **Socket** (p. 1900) instance that wraps the given **Socket** (p. 1900).*

  Exceptions

IOException	*if an I/O exception occurs while performing this operation.*
UnknownHostException (p. 2181)	*if the host is unknown.*

6.196.1 Detailed Description

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Since

1.0

6.196.2 Constructor & Destructor Documentation

6.196.2.1 decaf::internal::net::ssl::DefaultSSLSocketFactory::DefaultSSLSocketFactory (const std::string & *errorMessage*)

6.196.2.2 virtual decaf::internal::net::ssl::DefaultSSLSocketFactory::~DefaultSSLSocketFactory ( ) [virtual]

6.196.3 Member Function Documentation

6.196.3.1 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( ) [virtual]

Creates an unconnected **Socket** (p. 1900) object.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if the **Socket** (p. 1900) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 1917).

6.196.3.2 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const decaf::net::InetAddress * *host,* int *port*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1918).

6.196.3.3 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const decaf::net::InetAddress * *host,* int *port,* const decaf::net::InetAddress * *ifAddress,* int *localPort*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

The **Socket** (p. 1900) will be bound to the specified local address and port.

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1918).

6.196.3.4 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const std::string & *name,* int *port*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1919).

6.196.3.5 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const std::string & *name,* int *port,* const decaf::net::InetAddress * *ifAddress,* int *localPort*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1919).

6.196.3.6 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (decaf::net::Socket * *socket,* std::string *host,* int *port,* bool *autoClose*) [virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket	The existing socket to layer over.
host	The server host the original **Socket** (p. 1900) is connected to.
port	The server port the original **Socket** (p. 1900) is connected to.
autoClose	Should the layered over **Socket** (p. 1900) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 1900) instance that wraps the given **Socket** (p. 1900).

Exceptions

IOException	if an I/O exception occurs while performing this operation.
UnknownHostException *(p. 2181)*	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 1955).

6.196.3.7 virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getDefaultCipherSuites ( ) [virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 1956)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 1956).

6.196.3.8 virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getSupportedCipherSuites ( ) [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 1956)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 1956).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/**DefaultSSLSocketFactory.h**

6.197 activemq::transport::DefaultTransportListener Class Reference

A Utility class that create empty implementations for the **TransportListener** (p. 2176) interface so that a subclass only needs to override the one's its interested.

`#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.

6.197.1 Detailed Description

A Utility class that create empty implementations for the **TransportListener** (p. 2176) interface so that a subclass only needs to override the one's its interested.

6.197.2 Constructor & Destructor Documentation

6.197.2.1 virtual activemq::transport::DefaultTransportListener::~DefaultTransportListener ( ) [inline, virtual]

6.197.3 Member Function Documentation

6.197.3.1 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. 2161) deletes the command upon receipt.

Parameters

command	the received command object.

6.197.3.2 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.

6.197.3.3 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. 2178).

6.197.3.4 virtual void activemq::transport::DefaultTransportListener::transportResumed ( ) [inline, virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2178).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**DefaultTransportListener.h**

6.198 decaf::util::zip::Deflater Class Reference

This class compresses data using the *DEFLATE* algorithm (see specification).

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

Public Member Functions

- **Deflater** (int level, bool nowrap=false)

  Creates a new compressor using the specified compression level.
- **Deflater** ()

  Creates a new compressor with the default compression level.
- virtual ~**Deflater** ()
- void **setInput** (**const** unsigned char *buffer, int size, int offset, int length)

  Sets input data for compression.
- void **setInput** (**const** std::vector< unsigned char > &buffer, int offset, int length)

  Sets input data for compression.
- void **setInput** (**const** std::vector< unsigned char > &buffer)

  Sets input data for compression.
- void **setDictionary** (**const** unsigned char *buffer, int size, int offset, int length)

  Sets preset dictionary for compression.
- void **setDictionary** (**const** std::vector< unsigned char > &buffer, int offset, int length)

  Sets preset dictionary for compression.
- void **setDictionary** (**const** std::vector< unsigned char > &buffer)

  Sets preset dictionary for compression.
- void **setStrategy** (int strategy)

  Sets the compression strategy to the specified value.
- void **setLevel** (int level)

  Sets the compression level to the specified value.
- bool **needsInput** () **const**
- void **finish** ()

When called, indicates that compression should end with the current contents of the input buffer.

- bool **finished** () **const**
- int **deflate** (unsigned char *buffer, int size, int offset, int length)

  Fills specified buffer with compressed data.
- int **deflate** (std::vector< unsigned char > &buffer, int offset, int length)

  Fills specified buffer with compressed data.
- int **deflate** (std::vector< unsigned char > &buffer)

  Fills specified buffer with compressed data.
- long long **getAdler** () **const**
- long long **getBytesRead** () **const**
- long long **getBytesWritten** () **const**
- void **reset** ()

  Resets deflater so that a new set of input data can be processed.
- void **end** ()

  Closes the compressor and discards any unprocessed input.

Static Public Attributes

- static **const** int **BEST_SPEED**

  Compression level for fastest compression.
- static **const** int **BEST_COMPRESSION**

  Compression level for best compression.
- static **const** int **DEFAULT_COMPRESSION**

  Default compression level.
- static **const** int **DEFLATED**

  Compression method for the deflate algorithm (the only one currently supported).
- static **const** int **NO_COMPRESSION**

  Compression level for no compression.
- static **const** int **FILTERED**

  Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.
- static **const** int **HUFFMAN_ONLY**

  Compression strategy for Huffman coding only.
- static **const** int **DEFAULT_STRATEGY**

  Default compression strategy.

6.198.1 Detailed Description

This class compresses data using the *DEFLATE* algorithm (see `specification`).

Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **DeflaterOutputStream** (p. 921) and its descendants.

The typical usage of a **Deflater** (p. 913) instance outside this package consists of a specific call to one of its constructors before being passed to an instance of **DeflaterOutputStream** (p. 921).

See also

DeflaterOutputStream (p. 921)
Inflater (p. 1121)

Since

1.0

6.198.2 Constructor & Destructor Documentation

6.198.2.1 decaf::util::zip::Deflater::Deflater (int *level,* bool *nowrap* = `false`)

Creates a new compressor using the specified compression level.

If 'nowrap' is true then the ZLIB header and checksum fields will not be used in order to support the compression format used in both GZIP and PKZIP.

Parameters

level	The compression level to use (0-9).
nowrap	If true uses GZip compatible compression (defaults to false).

6.198.2.2 decaf::util::zip::Deflater::Deflater ( )

Creates a new compressor with the default compression level.

Compressed data will be generated in ZLIB format.

6.198.2.3 virtual decaf::util::zip::Deflater::~Deflater ( ) [virtual]

6.198.3 Member Function Documentation

6.198.3.1 int decaf::util::zip::Deflater::deflate (unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 917) should be called in order to determine if more input data is required.

Parameters

buffer	The Buffer to write the compressed data to.
size	The size of the passed buffer.
offset	The position in the Buffer to start writing at.
length	The maximum number of byte of data to write.

Returns

the actual number of bytes of compressed data.

Exceptions

NullPointerException	if buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.198.3.2 int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & *buffer,* int *offset,* int *length*)

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 917) should be called in order to determine if more input data is required.

Parameters

buffer	The Buffer to write the compressed data to.
offset	The position in the Buffer to start writing at.
length	The maximum number of byte of data to write.

Returns

the actual number of bytes of compressed data.

Exceptions

IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.198.3.3 int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & *buffer*)

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 917) should be called in order to determine if more input data is required.

Parameters

buffer	The Buffer to write the compressed data to.

Returns

the actual number of bytes of compressed data.

Exceptions

IllegalStateException	if in the end state.

6.198.3.4 void decaf::util::zip::Deflater::end ( )

Closes the compressor and discards any unprocessed input.

This method should be called when the compressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Deflater** (p. 913) object is undefined.

6.198.3.5 void decaf::util::zip::Deflater::finish ( )

When called, indicates that compression should end with the current contents of the input buffer.

6.198.3.6 bool decaf::util::zip::Deflater::finished ( ) const

Returns

true if the end of the compressed data output stream has been reached.

6.198.3.7 long long decaf::util::zip::Deflater::getAdler ( ) const

Returns

the ADLER-32 value of the uncompressed data.

Exceptions

IllegalStateException	if in the end state.

6.198.3.8 long long decaf::util::zip::Deflater::getBytesRead ( ) const

Returns

the total number of uncompressed bytes input so far.

Exceptions

IllegalStateException	if in the end state.

6.198.3.9 long long decaf::util::zip::Deflater::getBytesWritten ( ) const

Returns

the total number of compressed bytes output so far.

Exceptions

IllegalStateException	if in the end state.

6.198.3.10 bool decaf::util::zip::Deflater::needsInput ( ) const

Returns

true if the input data buffer is empty and **setInput()** (p. 919) should be called in order to provide more input

6.198.3.11 void decaf::util::zip::Deflater::reset ( )

Resets deflater so that a new set of input data can be processed.

Keeps current compression level and strategy settings.

Exceptions

IllegalStateException	if in the end state.

6.198.3.12 void decaf::util::zip::Deflater::setDictionary (const unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1124), **Inflater.getAdler()** (p. 1123) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

buffer	The buffer containing the preset dictionary.
size	The size of the passed dictionary buffer.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

NullPointerException	if buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.198.3.13 void decaf::util::zip::Deflater::setDictionary (const std::vector< unsigned char > & *buffer,* int *offset,* int *length*)

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1124), **Inflater.getAdler()** (p. 1123) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

buffer	The buffer containing the preset dictionary.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.198.3.14 void decaf::util::zip::Deflater::setDictionary (const std::vector< unsigned char > & *buffer*)

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1124), **Inflater.getAdler()** (p. 1123) can be called in order to get the Adler-32 value of the dictionary required for decompression.

Parameters

buffer	The buffer containing the preset dictionary.

Exceptions

IllegalStateException	if in the end state.

6.198.3.15 void decaf::util::zip::Deflater::setInput (const unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Sets input data for compression.

This should be called whenever **needsInput()** (p. 917) returns true indicating that more input data is required.

Parameters

buffer	The Buffer to read in for compression.
size	The size in bytes of the buffer passed.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

NullPointerException	if buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.198.3.16 void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*)

Sets input data for compression.

This should be called whenever **needsInput()** (p. 917) returns true indicating that more input data is required.

Parameters

buffer	The Buffer to read in for compression.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.198.3.17 void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & *buffer*)

Sets input data for compression.

This should be called whenever **needsInput()** (p. 917) returns true indicating that more input data is required.

Parameters

buffer	The Buffer to read in for compression.

Exceptions

IllegalStateException	if in the end state.

6.198.3.18 void decaf::util::zip::Deflater::setLevel (int *level*)

Sets the compression level to the specified value.

Parameters

level	The new Compression level to use.

Exceptions

IllegalArgumentException	if the level value is invalid.
IllegalStateException	if in the end state.

6.198.3.19 void decaf::util::zip::Deflater::setStrategy (int *strategy*)

Sets the compression strategy to the specified value.

Parameters

strategy	The new Compression strategy to use.

Exceptions

IllegalArgumentException	if the strategy value is invalid.
IllegalStateException	if in the end state.

6.198.4 Field Documentation

6.198.4.1 const int decaf::util::zip::Deflater::BEST_COMPRESSION [static]

Compression level for best compression.

6.198.4.2 const int decaf::util::zip::Deflater::BEST_SPEED [static]

Compression level for fastest compression.

6.198.4.3 const int decaf::util::zip::Deflater::DEFAULT_COMPRESSION [static]

Default compression level.

6.198.4.4 const int decaf::util::zip::Deflater::DEFAULT_STRATEGY [static]

Default compression strategy.

6.198.4.5 const int decaf::util::zip::Deflater::DEFLATED [static]

Compression method for the deflate algorithm (the only one currently supported).

6.198.4.6 const int decaf::util::zip::Deflater::FILTERED [static]

Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.

Forces more Huffman coding and less string matching.

6.198.4.7 const int decaf::util::zip::Deflater::HUFFMAN_ONLY [static]

Compression strategy for Huffman coding only.

6.198.4.8 const int decaf::util::zip::Deflater::NO_COMPRESSION [static]

Compression level for no compression.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Deflater.h**

6.199 decaf::util::zip::DeflaterOutputStream Class Reference

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

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

Inheritance diagram for decaf::util::zip::DeflaterOutputStream:

Public Member Functions

- **DeflaterOutputStream** (**decaf::io::OutputStream** ∗**outputStream**, bool **own**=false)

  *Creates a new DeflateOutputStream with a Default **Deflater** (p. 913) and buffer size.*
- **DeflaterOutputStream** (**decaf::io::OutputStream** ∗**outputStream**, **Deflater** ∗**deflater**, bool **own**=false, bool **ownDeflater**=false)

  *Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 913) and a default buffer size.*
- **DeflaterOutputStream** (**decaf::io::OutputStream** ∗**outputStream**, **Deflater** ∗**deflater**, int bufferSize, bool **own**=false, bool **ownDeflater**=false)

  *Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 913) and specified buffer size.*
- virtual ∼**DeflaterOutputStream** ()
- virtual void **finish** ()

  Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
  The object is generally no longer usable after calling close.

  Exceptions

IOException (p. 1198)	*if an error occurs while closing.*

  The default implementation of this method does nothing.
  *The close method of **FilterOutputStream** (p. 1037) calls its flush method, and then calls the close method of its underlying output stream.*

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (**const** unsigned char ∗buffer, int size, int offset, int length)
- virtual void **deflate** ()

  Writes a buffers worth of compressed data to the wrapped OutputStream.

Protected Attributes

- **Deflater** ∗ **deflater**

  *The **Deflater** (p. 913) for this stream.*
- std::vector< unsigned char > **buf**

The Buffer to use for.

- bool **ownDeflater**
- bool **isDone**

Static Protected Attributes

- static **const** std::size_t **DEFAULT_BUFFER_SIZE**

6.199.1 Detailed Description

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

Since

1.0

6.199.2 Constructor & Destructor Documentation

6.199.2.1 decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * *outputStream,* bool *own* = `false`)

Creates a new DeflateOutputStream with a Default **Deflater** (p. 913) and buffer size.

Parameters

outputStream	The OutputStream instance to wrap.
own	Should this filter take ownership of the OutputStream pointer (default is false).

6.199.2.2 decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * *outputStream,* Deflater * *deflater,* bool *own* = `false`, bool *ownDeflater* = `false`)

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 913) and a default buffer size.

When the user supplied a **Deflater** (p. 913) instance the DeflaterOutpotStream does not take ownership of the **Deflater** (p. 913) pointer unless the ownDeflater parameter is set to true, the caller is still responsible for deleting the **Deflater** (p. 913) when ownDeflater is false.

Parameters

outputStream	The OutputStream instance to wrap.
deflater	The user supplied **Deflater** (p. 913) to use for compression. (
own	Should this filter take ownership of the OutputStream pointer (default is false).
ownDeflater	Should the filter take ownership of the passed **Deflater** (p. 913) object (default is false).

Exceptions

NullPointerException	if the **Deflater** (p. 913) given is NULL.

6.199.2.3 decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream * *outputStream,* Deflater * *deflater,* int *bufferSize,* bool *own* = `false`, bool *ownDeflater* = `false`)

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 913) and specified buffer size.

When the user supplied a **Deflater** (p. 913) instance the DeflaterOutpotStream does not take ownership of the

Deflater (p. 913) pointer unless the ownDeflater parameter is set to true, otherwise the caller is still responsible for deleting the **Deflater** (p. 913).

Parameters

outputStream	The OutputStream instance to wrap.
deflater	The user supplied **Deflater** (p. 913) to use for compression.
bufferSize	The size of the input buffer.
own	Should this filter take ownership of the OutputStream pointer (default is false).
ownDeflater	Should the filter take ownership of the passed **Deflater** (p. 913) object (default is false).

Exceptions

NullPointerException	if the **Deflater** (p. 913) given is NULL.
IllegalArgumentException	if bufferSize is 0.

6.199.2.4 virtual decaf::util::zip::DeflaterOutputStream::~DeflaterOutputStream ( ) [virtual]

6.199.3 Member Function Documentation

6.199.3.1 virtual void decaf::util::zip::DeflaterOutputStream::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(p. 1198)*	if an error occurs while closing.

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1037) calls its flush method, and then calls the close method of its underlying output stream.

Finishes writing any remaining data to the OutputStream then closes the stream.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.199.3.2 virtual void decaf::util::zip::DeflaterOutputStream::deflate ( ) [protected, virtual]

Writes a buffers worth of compressed data to the wrapped OutputStream.

6.199.3.3 virtual void decaf::util::zip::DeflaterOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.199.3.4 virtual void decaf::util::zip::DeflaterOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.199.3.5 virtual void decaf::util::zip::DeflaterOutputStream::finish ( ) [virtual]

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

Exceptions

IOException	if an I/O error occurs.

6.199.4 Field Documentation

6.199.4.1 std::vector<unsigned char> decaf::util::zip::DeflaterOutputStream::buf [protected]

The Buffer to use for.

6.199.4.2 const std::size_t decaf::util::zip::DeflaterOutputStream::DEFAULT_BUFFER_SIZE [static, protected]

6.199.4.3 Deflater* decaf::util::zip::DeflaterOutputStream::deflater [protected]

The **Deflater** (p. 913) for this stream.

6.199.4.4 bool decaf::util::zip::DeflaterOutputStream::isDone [protected]

6.199.4.5 bool decaf::util::zip::DeflaterOutputStream::ownDeflater [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**DeflaterOutputStream.h**

6.200 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.

6.200.1 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.

6.200.2 Constructor & Destructor Documentation

6.200.2.1 virtual decaf::util::concurrent::Delayed::~Delayed ( ) [inline, virtual]

6.200.3 Member Function Documentation

6.200.3.1 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**

6.201 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. 1426) Delivery Mode.*

Public Member Functions

- virtual ~**DeliveryMode** ()

6.201.1 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. 1426) it can mark the **Message** (p. 1426) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 1426) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 1426) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 1426) throughput.

The **DeliveryMode** (p. 925) covers only the transport of the **Message** (p. 1426) for sending client to its destination and doesn't apply to the receiving **Message** (p. 1426) consumer. The receiving Consumer can drop **Message** (p. 1426)'s based on configuration such as memory limits or **Message** (p. 1426) 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. 1426) consumer allows for it.

Since

1.0

6.201.2 Member Enumeration Documentation

6.201.2.1 enum cms::DeliveryMode::DELIVERY_MODE

Enumeration values for **Message** (p. 1426) Delivery Mode.

Enumerator:

PERSISTENT

NON_PERSISTENT

6.201.3 Constructor & Destructor Documentation

6.201.3.1 virtual cms::DeliveryMode::~DeliveryMode ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**DeliveryMode.h**

6.202 decaf::util::Deque< E > Class Template Reference

Defines a 'Double ended **Queue** (p. 1723)' interface that allows for insertion and removal of elements from both ends.

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

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

Public Member Functions

- virtual ~**Deque** ()
- virtual void **addFirst** (**const** E &**element**)=0

  *Inserts an element onto the front of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.*
- virtual void **addLast** (**const** E &**element**)=0

  *Inserts an element onto the end of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.*
- virtual bool **offerFirst** (**const** E &**element**)=0

  *This method attempts to insert the given element into the **Deque** (p. 926) at the front end.*
- virtual bool **offerLast** (**const** E &**element**)=0

  *This method attempts to insert the given element into the **Deque** (p. 926) at the end.*
- virtual E **removeFirst** ()=0

  *Removes the topmost element from the **Deque** (p. 926) and returns it.*
- virtual E **removeLast** ()=0

  *Removes the last element from the **Deque** (p. 926) and returns it.*
- virtual bool **pollFirst** (E &**element**)=0

  *Removes the first element from the **Deque** (p. 926) assigns it to the element reference passed.*
- virtual bool **pollLast** (E &**element**)=0

  *Removes the last element from the **Deque** (p. 926) assigns it to the element reference passed.*
- virtual E & **getFirst** ()=0

  *Attempts to fetch a reference to the first element in the **Deque** (p. 926).*

- virtual **const** E & **getFirst** () **const** =0
- virtual E & **getLast** ()=0

  *Attempts to fetch a reference to the last element in the **Deque** (p. 926).*
- virtual **const** E & **getLast** () **const** =0
- virtual bool **peekFirst** (E &value) **const** =0

  *Retrieves the first element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).*
- virtual bool **peekLast** (E &value) **const** =0

  *Retrieves the last element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).*
- virtual bool **removeFirstOccurrence** (**const** E &value)=0

  *Removes the first occurrence of the specified element from this **Deque** (p. 926).*
- virtual bool **removeLastOccurrence** (**const** E &value)=0

  *Removes the last occurrence of the specified element from this **Deque** (p. 926).*
- virtual void **push** (**const** E &**element**)=0

  Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an IllegalStateException if no space is currently available.
- virtual E **pop** ()=0

  *Treats this **Deque** (p. 926) as a stack and attempts to pop an element off the top.*
- virtual **Iterator**< E > * **descendingIterator** ()=0

  *Provides an **Iterator** (p. 1209) over this **Collection** (p. 660) that traverses the element in reverse order.*
- virtual **Iterator**< E > * **descendingIterator** () **const** =0

6.202.1 Detailed Description

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

Defines a 'Double ended **Queue** (p. 1723)' interface that allows for insertion and removal of elements from both ends.

Generally there is no limit on the number of elements that can be placed into a **Deque** (p. 926).

Unlike a **List** (p. 1286) the **Deque** (p. 926) doesn't provide index element based access, however methods are provided to grant access to interior elements.

Since

1.0

6.202.2 Constructor & Destructor Documentation

6.202.2.1 template<typename E> virtual decaf::util::Deque< E >::~Deque ( ) [inline, virtual]

6.202.3 Member Function Documentation

6.202.3.1 template<typename E> virtual void decaf::util::Deque< E >::addFirst (const E & *element*) [pure virtual]

Inserts an element onto the front of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 926) it is preferable to call offerFirst instead.

Parameters

element	The element to be placed at the front of the **Deque** (p. 926).

Exceptions

IllegalStateException	if the element cannot be added at this time due to capacity restrictions
NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implemented in **decaf::util::LinkedList< E >** (p. 1274), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1274), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1274), **decaf::util::LinkedList< CompositeTask * >** (p. 1274), **decaf::util::LinkedList< URI >** (p. 1274), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1274), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1274), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1274), **decaf::util::LinkedList< Pointer< Command > >** (p. 1274), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1274), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1274), **decaf::util::LinkedList< cms::Destination * >** (p. 1274), **decaf::util::LinkedList< cms::Session * >** (p. 1274), and **decaf::util::LinkedList< cms::Connection * >** (p. 1274).

6.202.3.2 template<typename E> virtual void decaf::util::Deque< E >::addLast (const E & *element*) [pure virtual]

Inserts an element onto the end of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 926) it is preferable to call offerLast instead.

Parameters

element	The element to be placed at the end of the **Deque** (p. 926).

Exceptions

IllegalStateException	if the element cannot be added at this time due to capacity restrictions
NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implemented in **decaf::util::LinkedList< E >** (p. 1274), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1274), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1274), **decaf::util::LinkedList< CompositeTask * >** (p. 1274), **decaf::util::LinkedList< URI >** (p. 1274), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1274), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1274), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1274), **decaf::util::LinkedList< Pointer< Command > >** (p. 1274), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1274), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1274), **decaf::util::LinkedList< cms::Destination * >** (p. 1274), **decaf::util::LinkedList< cms::Session * >** (p. 1274), and **decaf::util::LinkedList< cms::Connection * >** (p. 1274).

6.202.3.3 template<typename E> virtual Iterator<E>* decaf::util::Deque< E >::descendingIterator ( ) [pure virtual]

Provides an **Iterator** (p. 1209) over this **Collection** (p. 660) that traverses the element in reverse order.

Returns

a new **Iterator** (p. 1209) instance that moves from last to first.

Implemented in **decaf::util::LinkedList< E >** (p. 1275), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1275), **decaf::util::LinkedList< CompositeTask * >** (p. 1275), **decaf::util::LinkedList< URI >** (p. 1275), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1275), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1275), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1275), **decaf::util::LinkedList< Pointer< Command > >**

(p. 1275), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1275), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1275), **decaf::util::LinkedList< cms::Destination * >** (p. 1275), **decaf::util::LinkedList< cms::Session * >** (p. 1275), and **decaf::util::LinkedList< cms::Connection * >** (p. 1275).

6.202.3.4 template<typename E> virtual Iterator<E>* decaf::util::Deque< E >::descendingIterator ( ) const [pure virtual]

Implemented in **decaf::util::LinkedList< E >** (p. 1276), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1276), **decaf::util::LinkedList< CompositeTask * >** (p. 1276), **decaf::util::LinkedList< URI >** (p. 1276), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1276), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1276), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1276), **decaf::util::LinkedList< Pointer< Command > >** (p. 1276), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1276), **decaf::util::LinkedList< cms::Destination * >** (p. 1276), **decaf::util::LinkedList< cms::Session * >** (p. 1276), and **decaf::util::LinkedList< cms::Connection * >** (p. 1276).

6.202.3.5 template<typename E> virtual E& decaf::util::Deque< E >::getFirst ( ) [pure virtual]

Attempts to fetch a reference to the first element in the **Deque** (p. 926).

This method does not remove the element from the **Deque** (p. 926) but simply returns a reference to it.

Returns

reference to the first element in the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implemented in **decaf::util::LinkedList< E >** (p. 1277), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1277), **decaf::util::LinkedList< CompositeTask * >** (p. 1277), **decaf::util::LinkedList< URI >** (p. 1277), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1277), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1277), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1277), **decaf::util::LinkedList< Pointer< Command > >** (p. 1277), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1277), **decaf::util::LinkedList< cms::Destination * >** (p. 1277), **decaf::util::LinkedList< cms::Session * >** (p. 1277), and **decaf::util::LinkedList< cms::Connection * >** (p. 1277).

6.202.3.6 template<typename E> virtual const E& decaf::util::Deque< E >::getFirst ( ) const [pure virtual]

Implemented in **decaf::util::LinkedList< E >** (p. 1277), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1277), **decaf::util::LinkedList< CompositeTask * >** (p. 1277), **decaf::util::LinkedList< URI >** (p. 1277), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1277), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1277), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1277), **decaf::util::LinkedList< Pointer< Command > >** (p. 1277), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1277), **decaf::util::LinkedList< cms::Destination * >** (p. 1277), **decaf::util::LinkedList< cms::Session * >** (p. 1277), and **decaf::util::LinkedList< cms::Connection * >** (p. 1277).

6.202.3.7 template<typename E> virtual E& decaf::util::Deque< E >::getLast ( ) [pure virtual]

Attempts to fetch a reference to the last element in the **Deque** (p. 926).

This method does not remove the element from the **Deque** (p. 926) but simply returns a reference to it.

Returns

reference to the last element in the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implemented in **decaf::util::LinkedList< E >** (p. 1277), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1277), **decaf::util::LinkedList< CompositeTask * >** (p. 1277), **decaf::util::LinkedList< URI >** (p. 1277), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1277), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1277), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1277), **decaf::util::LinkedList< Pointer< Command > >** (p. 1277), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1277), **decaf::util::LinkedList< cms::Destination * >** (p. 1277), **decaf::util::LinkedList< cms::Session * >** (p. 1277), and **decaf::util::LinkedList< cms::Connection * >** (p. 1277).

6.202.3.8 template<typename E> virtual const E& decaf::util::Deque< E >::getLast ( ) const [pure virtual]

Implemented in **decaf::util::LinkedList< E >** (p. 1277), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1277), **decaf::util::LinkedList< CompositeTask * >** (p. 1277), **decaf::util::LinkedList< URI >** (p. 1277), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1277), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1277), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1277), **decaf::util::LinkedList< Pointer< Command > >** (p. 1277), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1277), **decaf::util::LinkedList< cms::Destination * >** (p. 1277), **decaf::util::LinkedList< cms::Session * >** (p. 1277), and **decaf::util::LinkedList< cms::Connection * >** (p. 1277).

6.202.3.9 template<typename E> virtual bool decaf::util::Deque< E >::offerFirst (const E & *element*) [pure virtual]

This method attempts to insert the given element into the **Deque** (p. 926) at the front end.

Unlike the addFirst method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

element	The element to add to this **Deque** (p. 926).

Returns

true if the element was added, false otherwise.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implemented in **decaf::util::LinkedList< E >** (p. 1279), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1279), **decaf::util::LinkedList< CompositeTask * >** (p. 1279), **decaf::util::LinkedList< URI >** (p. 1279), **decaf::util::LinkedList< Pointer<**

MessageDispatch > > (p. 1279), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1279), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1279), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1279), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1279), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1279), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1279), **decaf::util::LinkedList**< **cms::Session** * > (p. 1279), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1279).

6.202.3.10 template<typename E> virtual bool decaf::util::Deque< E >::offerLast (const E & *element*) [pure virtual]

This method attempts to insert the given element into the **Deque** (p. 926) at the end.

Unlike the addLast method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

element	The element to add to this **Deque** (p. 926).

Returns

true if the element was added, false otherwise.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implemented in **decaf::util::LinkedList**< **E** > (p. 1280), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1280), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1280), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1280), **decaf::util::LinkedList**< **URI** > (p. 1280), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1280), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1280), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1280), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1280), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1280), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1280), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1280), **decaf::util::LinkedList**< **cms::Session** * > (p. 1280), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1280).

6.202.3.11 template<typename E> virtual bool decaf::util::Deque< E >::peekFirst (E & *value*) const [pure virtual]

Retrieves the first element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).

If this call is successful it returns true. Unlike getFirst this method does not throw an exception if the **Deque** (p. 926) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implemented in **decaf::util::LinkedList**< **E** > (p. 1281), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1281), **decaf::util::LinkedList**< **cms::MessageConsumer** * > (p. 1281), **decaf::util::LinkedList**< **CompositeTask** * > (p. 1281), **decaf::util::LinkedList**< **URI** > (p. 1281), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1281), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1281), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1281), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1281), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1281), **decaf::util::LinkedList**< **cms::MessageProducer** * > (p. 1281), **decaf::util::LinkedList**< **cms::Destination** * > (p. 1281), **decaf::util::LinkedList**< **cms::Session** * > (p. 1281), and **decaf::util::LinkedList**< **cms::Connection** * > (p. 1281).

6.202.3.12 template<typename E> virtual bool decaf::util::Deque< E >::peekLast (E & *value*) const [pure virtual]

Retrieves the last element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).

If this call is successful it returns true. Unlike getLast this method does not throw an exception if the **Deque** (p. 926) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implemented in **decaf::util::LinkedList< E >** (p. 1281), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1281), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1281), **decaf::util::LinkedList< CompositeTask * >** (p. 1281), **decaf::util::LinkedList< URI >** (p. 1281), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1281), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1281), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1281), **decaf::util::LinkedList< Pointer< Command > >** (p. 1281), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1281), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1281), **decaf::util::LinkedList< cms::Destination * >** (p. 1281), **decaf::util::LinkedList< cms::Session * >** (p. 1281), and **decaf::util::LinkedList< cms::Connection * >** (p. 1281).

6.202.3.13 template<typename E> virtual bool decaf::util::Deque< E >::pollFirst (E & *element*) [pure virtual]

Removes the first element from the **Deque** (p. 926) assigns it to the element reference passed.

Parameters

element	Reference to an variable that can be assigned the value of the head of this **Deque** (p. 926).

Returns

true if an element was available to remove, false otherwise.

Implemented in **decaf::util::LinkedList< E >** (p. 1281), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1281), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1281), **decaf::util::LinkedList< CompositeTask * >** (p. 1281), **decaf::util::LinkedList< URI >** (p. 1281), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1281), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1281), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1281), **decaf::util::LinkedList< Pointer< Command > >** (p. 1281), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1281), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1281), **decaf::util::LinkedList< cms::Destination * >** (p. 1281), **decaf::util::LinkedList< cms::Session * >** (p. 1281), and **decaf::util::LinkedList< cms::Connection * >** (p. 1281).

6.202.3.14 template<typename E> virtual bool decaf::util::Deque< E >::pollLast (E & *element*) [pure virtual]

Removes the last element from the **Deque** (p. 926) assigns it to the element reference passed.

Parameters

element	Reference to an variable that can be assigned the value of the tail of this **Deque** (p. 926).

Returns

true if an element was available to remove, false otherwise.

Implemented in **decaf::util::LinkedList< E >** (p. 1282), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1282), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1282), **decaf::util::LinkedList< CompositeTask * >** (p. 1282), **decaf::util::LinkedList< URI >** (p. 1282), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1282), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1282), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1282), **decaf::util::LinkedList< Pointer< Command > >** (p. 1282), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1282), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1282), **decaf::util::LinkedList< cms::Destination * >** (p. 1282), **decaf::util::LinkedList< cms::Session * >** (p. 1282), and **decaf::util::LinkedList< cms::Connection * >** (p. 1282).

6.202.3.15 template<typename E> virtual E decaf::util::Deque< E >::pop ( ) [pure virtual]

Treats this **Deque** (p. 926) as a stack and attempts to pop an element off the top.

If there's no element to pop then a NuSuchElementException is thrown, otherwise the top element is removed and assigned to the reference passed.

This operation performs the same basic function as the removeFirst method.

Returns

the element at the front of this deque which would be the top of a stack.

Exceptions

NoSuchElementException *(p. 1537)*	if there is nothing on the top of the stack.

Implemented in **decaf::util::LinkedList< E >** (p. 1282), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1282), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1282), **decaf::util::LinkedList< CompositeTask * >** (p. 1282), **decaf::util::LinkedList< URI >** (p. 1282), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1282), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1282), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1282), **decaf::util::LinkedList< Pointer< Command > >** (p. 1282), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1282), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1282), **decaf::util::LinkedList< cms::Destination * >** (p. 1282), **decaf::util::LinkedList< cms::Session * >** (p. 1282), and **decaf::util::LinkedList< cms::Connection * >** (p. 1282).

6.202.3.16 template<typename E> virtual void decaf::util::Deque< E >::push (const E & *element*) [pure virtual]

Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an IllegalStateException if no space is currently available.

This method performs the same basic operation as the addFirst method.

Parameters

element	The element to be pushed onto the **Deque** (p. 926).

Exceptions

IllegalStateException	if the element cannot be added at this time due to capacity restrictions
NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implemented in **decaf::util::LinkedList< E >** (p. 1282), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1282), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1282), **decaf::util::LinkedList< CompositeTask * >** (p. 1282), **decaf::util::LinkedList< URI >** (p. 1282), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1282), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1282), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1282), **decaf::util::LinkedList< Pointer< Command > >** (p. 1282), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1282), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1282), **decaf::util::LinkedList< cms::Destination * >** (p. 1282), **decaf::util::LinkedList< cms::Session * >** (p. 1282), and **decaf::util::LinkedList< cms::Connection * >** (p. 1282).

6.202.3.17 template<typename E> virtual E decaf::util::Deque< E >::removeFirst ( ) [pure virtual]

Removes the topmost element from the **Deque** (p. 926) and returns it.

Unlike the pollFirst method this method throws a NuSuchElementException if the **Deque** (p. 926) is empty.

Returns

the element at the Head of the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implemented in **decaf::util::LinkedList< E >** (p. 1284), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1284), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1284), **decaf::util::LinkedList< CompositeTask * >** (p. 1284), **decaf::util::LinkedList< URI >** (p. 1284), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1284), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1284), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1284), **decaf::util::LinkedList< Pointer< Command > >** (p. 1284), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1284), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1284), **decaf::util::LinkedList< cms::Destination * >** (p. 1284), **decaf::util::LinkedList< cms::Session * >** (p. 1284), and **decaf::util::LinkedList< cms::Connection * >** (p. 1284).

6.202.3.18 template<typename E> virtual bool decaf::util::Deque< E >::removeFirstOccurrence (const E & *value*) [pure virtual]

Removes the first occurrence of the specified element from this **Deque** (p. 926).

If there is no matching element then the **Deque** (p. 926) is left unchanged.

Parameters

value	The value to be removed from this **Deque** (p. 926).

Returns

true if the **Deque** (p. 926) was modified as a result of this operation.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.

Implemented in **decaf::util::LinkedList< E >** (p. 1284), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1284), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1284), **decaf::util::LinkedList< CompositeTask * >** (p. 1284), **decaf::util::LinkedList< URI >** (p. 1284), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1284), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1284), **decaf-**

::util::LinkedList< PrimitiveValueNode > (p. 1284), **decaf::util::LinkedList< Pointer< Command > >** (p. 1284), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1284), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1284), **decaf::util::LinkedList< cms::Destination * >** (p. 1284), **decaf::util::LinkedList< cms::Session * >** (p. 1284), and **decaf::util::LinkedList< cms::Connection * >** (p. 1284).

6.202.3.19 template<typename E> virtual E decaf::util::Deque< E >::removeLast ( ) [pure virtual]

Removes the last element from the **Deque** (p. 926) and returns it.

Unlike the pollLast method this method throws a NuSuchElementException if the **Deque** (p. 926) is empty.

Returns

the element at the Tail of the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implemented in **decaf::util::LinkedList< E >** (p. 1284), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1284), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1284), **decaf::util::LinkedList< CompositeTask * >** (p. 1284), **decaf::util::LinkedList< URI >** (p. 1284), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1284), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1284), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1284), **decaf::util::LinkedList< Pointer< Command > >** (p. 1284), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1284), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1284), **decaf::util::LinkedList< cms::Destination * >** (p. 1284), **decaf::util::LinkedList< cms::Session * >** (p. 1284), and **decaf::util::LinkedList< cms::Connection * >** (p. 1284).

6.202.3.20 template<typename E> virtual bool decaf::util::Deque< E >::removeLastOccurrence (const E & *value*) [pure virtual]

Removes the last occurrence of the specified element from this **Deque** (p. 926).

If there is no matching element then the **Deque** (p. 926) is left unchanged.

Parameters

value	The value to be removed from this **Deque** (p. 926).

Returns

true if the **Deque** (p. 926) was modified as a result of this operation.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.

Implemented in **decaf::util::LinkedList< E >** (p. 1285), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1285), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1285), **decaf::util::LinkedList< CompositeTask * >** (p. 1285), **decaf::util::LinkedList< URI >** (p. 1285), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1285), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1285), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1285), **decaf::util::LinkedList< Pointer< Command > >** (p. 1285), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1285), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1285), **decaf::util::LinkedList< cms::Destination * >** (p. 1285), **decaf::util::LinkedList< cms::Session * >** (p. 1285), and **decaf::util::LinkedList< cms::Connection * >** (p. 1285).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Deque.h**

6.203 cms::Destination Class Reference

A **Destination** (p. 936) 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** () throw ()
- virtual **DestinationType getDestinationType** () **const** =0

  *Retrieve the **Destination** (p. 936) Type for this **Destination** (p. 936).*
- 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. 936) object to this one.*
- virtual bool **equals** (**const cms::Destination** &other) **const** =0

  *Compares two **Destination** (p. 936) instances to determine if they represent the same logic **Destination** (p. 936).*
- virtual **const CMSProperties** & **getCMSProperties** () **const** =0

  Retrieve any properties that might be part of the destination that was specified.

6.203.1 Detailed Description

A **Destination** (p. 936) object encapsulates a provider-specific address.

There is no standard definition of a **Destination** (p. 936) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 936) address.

All CMS **Destination** (p. 936) objects support concurrent use.

Since

1.0

6.203.2 Member Enumeration Documentation

6.203.2.1 enum cms::Destination::DestinationType

Enumerator:

TOPIC

QUEUE

TEMPORARY_TOPIC

TEMPORARY_QUEUE

6.203.3 Constructor & Destructor Documentation

6.203.3.1 virtual cms::Destination::~Destination ( ) throw () [virtual]

6.203.4 Member Function Documentation

6.203.4.1 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. 250), **activemq::commands::ActiveMQTopic** (p. 322), **activemq::commands::ActiveMQTempQueue** (p. 300), and **activemq::commands::ActiveMQTempTopic** (p. 308).

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setCMSDestination(), and activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setCMSReplyTo().

6.203.4.2 virtual void cms::Destination::copy (const cms::Destination & *source*) [pure virtual]

Copies the contents of the given **Destination** (p. 936) object to this one.

Parameters

source	The source **Destination** (p. 936) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 250), **activemq::commands::ActiveMQTopic** (p. 322), **activemq::commands::ActiveMQTempQueue** (p. 301), and **activemq::commands::ActiveMQTempTopic** (p. 308).

6.203.4.3 virtual bool cms::Destination::equals (const cms::Destination & *other*) const [pure virtual]

Compares two **Destination** (p. 936) instances to determine if they represent the same logic **Destination** (p. 936).

Parameters

other	The other destination to compare this one to.

Returns

true if the two destinations are the same.

Implemented in **activemq::commands::ActiveMQQueue** (p. 251), **activemq::commands::ActiveMQTopic** (p. 323), **activemq::commands::ActiveMQTempQueue** (p. 302), and **activemq::commands::ActiveMQTempTopic** (p. 309).

6.203.4.4 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. 644) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 251), **activemq::commands::ActiveMQTopic** (p. 323), **activemq::commands::ActiveMQTempQueue** (p. 302), and **activemq::commands::ActiveMQTempTopic** (p. 309).

6.203.4.5 virtual DestinationType cms::Destination::getDestinationType ( ) const [pure virtual]

Retrieve the **Destination** (p. 936) Type for this **Destination** (p. 936).

Returns

The **Destination** (p. 936) Type

Implemented in **activemq::commands::ActiveMQQueue** (p. 251), **activemq::commands::ActiveMQTopic** (p. 324), **activemq::commands::ActiveMQTempQueue** (p. 303), and **activemq::commands::ActiveMQTempTopic** (p. 310).

The documentation for this class was generated from the following file:

- src/main/cms/**Destination.h**

6.204 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**

6.204.1 Field Documentation

6.204.1.1 const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_CHILD [static]

6.204.1.2 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**

6.205 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **DestinationInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **ActiveMQDestination** > **destination**
- unsigned char **operationType**
- long long **timeout**
- std::vector
  < **decaf::lang::Pointer**
  < **BrokerId** > > **brokerPath**

6.205.1 Constructor & Destructor Documentation

6.205.1.1 activemq::commands::DestinationInfo::DestinationInfo ( )

6.205.1.2 virtual activemq::commands::DestinationInfo::~DestinationInfo ( ) [virtual]

6.205.2 Member Function Documentation

6.205.2.1 virtual DestinationInfo* activemq::commands::DestinationInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.205.2.2 virtual void activemq::commands::DestinationInfo::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. 382).

6.205.2.3 virtual bool activemq::commands::DestinationInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.205.2.4 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ( ) const [virtual]

6.205.2.5 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ( ) [virtual]

6.205.2.6 virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ( ) const [virtual]

6.205.2.7 virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ( ) [virtual]

6.205.2.8 virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.205.2.9 virtual const Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination ( ) const [virtual]

Referenced by activemq::state::ConnectionState::removeTempDestination().

6.205.2.10 virtual Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination ( ) [virtual]

6.205.2.11 virtual unsigned char activemq::commands::DestinationInfo::getOperationType ( ) const [virtual]

6.205.2.12 virtual long long activemq::commands::DestinationInfo::getTimeout ( ) const [virtual]

6.205.2.13 virtual void activemq::commands::DestinationInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.205.2.14 virtual void activemq::commands::DestinationInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.205.2.15 virtual void activemq::commands::DestinationInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.205.2.16 virtual void activemq::commands::DestinationInfo::setOperationType (unsigned char *operationType*) [virtual]

6.205.2.17 virtual void activemq::commands::DestinationInfo::setTimeout (long long *timeout*) [virtual]

6.205.2.18 virtual std::string activemq::commands::DestinationInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.205.2.19 virtual Pointer<Command> activemq::commands::DestinationInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.205.3 Field Documentation

6.205.3.1 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::DestinationInfo::brokerPath [protected]

6.205.3.2 Pointer<ConnectionId> activemq::commands::DestinationInfo::connectionId [protected]

6.205.3.3 Pointer<ActiveMQDestination> activemq::commands::DestinationInfo::destination [protected]

6.205.3.4 const unsigned char activemq::commands::DestinationInfo::ID_DESTINATIONINFO = 8 [static]

6.205.3.5 unsigned char activemq::commands::DestinationInfo::operationType [protected]

6.205.3.6 long long activemq::commands::DestinationInfo::timeout [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DestinationInfo.h**

6.206 activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 943).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/DestinationInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual ~**DestinationInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.206.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 943).

NOTE!: This file is 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::generated::DestinationInfoMarshaller::DestinationInfoMarshaller ( ) [inline]

6.206.2.2 virtual activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::∼DestinationInfoMarshaller ( ) [inline, virtual]

6.206.3 Member Function Documentation

6.206.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.206.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.206.3.3 virtual void activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::looseMarshal (OpenWireFormat ∗ *format*, commands::DataStructure ∗ *command*, decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.206.3.4 virtual void activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.206.3.5 virtual int activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.206.3.6 virtual void activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.206.3.7 virtual void activemq::wireformat::openwire::marshal::generated::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**DestinationInfoMarshaller.h**

6.207 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** () throw ()
- 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

  Resolves the given name to a destination.

6.207.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.207.2 Constructor & Destructor Documentation

6.207.2.1 virtual activemq::cmsutil::DestinationResolver::~DestinationResolver ( ) throw () [inline, virtual]

6.207.3 Member Function Documentation

6.207.3.1 virtual void activemq::cmsutil::DestinationResolver::destroy ( ) [pure virtual]

Destroys any allocated resources.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 985).

6.207.3.2 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. 946)).

Parameters

mgr	the resource lifecycle manager.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 985).

6.207.3.3 virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName (cms::Session * *session,* const std::string & *destName,* bool *pubSubDomain*) [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. 640)*	if resolution failed.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 985).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DestinationResolver.h**

6.208 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always destroys the oldest unexecuted task in the **Queue** (p. 1723) and then attempts to execute the rejected task using the passed in executor.

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

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy:

Public Member Functions

- **DiscardOldestPolicy** ()
- virtual ~**DiscardOldestPolicy** ()
- virtual void **rejectedExecution** (**decaf::lang::Runnable** *task, **ThreadPoolExecutor** *executer)

  *Method that may be invoked by a **ThreadPoolExecutor** (p. 2104) when **execute** (p. 2110) cannot accept a task.*

6.208.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always destroys the oldest unexecuted task in the **Queue** (p. 1723) and then attempts to execute the rejected task using the passed in executor.

Since

1.0

6.208.2 Constructor & Destructor Documentation

6.208.2.1 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::DiscardOldestPolicy ( ) [inline]

6.208.2.2 virtual decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::~DiscardOldestPolicy ( ) [inline, virtual]

6.208.3 Member Function Documentation

6.208.3.1 virtual void decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::rejectedExecution (decaf::lang::Runnable * *r*, ThreadPoolExecutor * *executer*) [inline, virtual]

Method that may be invoked by a **ThreadPoolExecutor** (p. 2104) when **execute** (p. 2110) cannot accept a task.

This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p. 1004).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p. 1755), which will be propagated to the caller of **execute** (p. 2110).

Parameters

r	The pointer to the runnable task requested to be executed.
executor	The pointer to the executor attempting to execute this task.

Exceptions

RejectedExecution-Exception *(p. 1755)*	if there is no remedy.

Implements **decaf::util::concurrent::RejectedExecutionHandler** (p. 1757).

References decaf::util::concurrent::ThreadPoolExecutor::execute(), decaf::util::concurrent::ThreadPoolExecutor::getQueue(), and decaf::util::concurrent::ThreadPoolExecutor::isShutdown().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.209 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy Class Reference

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always destroys the rejected task and returns quietly.

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

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy:

Public Member Functions

- **DiscardPolicy** ()
- virtual ∼**DiscardPolicy** ()
- virtual void **rejectedExecution** (**decaf::lang::Runnable** ∗task, **ThreadPoolExecutor** ∗executer **DECAF_UNUSED**)

6.209.1 Detailed Description

Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always destroys the rejected task and returns quietly.

Since

1.0

6.209.2 Constructor & Destructor Documentation

6.209.2.1 decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy::DiscardPolicy ( ) [inline]

6.209.2.2 virtual decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy::∼DiscardPolicy ( ) [inline, virtual]

6.209.3 Member Function Documentation

6.209.3.1 virtual void **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy-::rejectedExecution (decaf::lang::Runnable** ∗ ***task,*** **ThreadPoolExecutor** ∗**executer** ***DECAF_UNUSED***) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.210 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 **DataStructure*** (p. 877) *Type as defined in CommandTypes.h.*
- virtual **DiscoveryEvent** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) *such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure*** (p. 877) *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 Attributes

- std::string **serviceName**
- std::string **brokerName**

6.210.1 Constructor & Destructor Documentation

6.210.1.1 activemq::commands::DiscoveryEvent::DiscoveryEvent ( )

6.210.1.2 virtual activemq::commands::DiscoveryEvent::~DiscoveryEvent ( ) [virtual]

6.210.2 Member Function Documentation

6.210.2.1 virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.210.2.2 virtual void activemq::commands::DiscoveryEvent::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. 879).

6.210.2.3 virtual bool activemq::commands::DiscoveryEvent::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.210.2.4 virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName ( ) const [virtual]

6.210.2.5 virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName ( ) [virtual]

6.210.2.6 virtual unsigned char activemq::commands::DiscoveryEvent::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.210.2.7 virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName ( ) const [virtual]

6.210.2.8 virtual std::string& activemq::commands::DiscoveryEvent::getServiceName ( ) [virtual]

6.210.2.9 virtual void activemq::commands::DiscoveryEvent::setBrokerName (const std::string & *brokerName*) [virtual]

6.210.2.10 virtual void activemq::commands::DiscoveryEvent::setServiceName (const std::string & *serviceName*) [virtual]

6.210.2.11 virtual std::string activemq::commands::DiscoveryEvent::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.210.3 Field Documentation

6.210.3.1 std::string activemq::commands::DiscoveryEvent::brokerName [protected]

6.210.3.2 const unsigned char activemq::commands::DiscoveryEvent::ID_DISCOVERYEVENT = 40 [static]

6.210.3.3 std::string activemq::commands::DiscoveryEvent::serviceName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DiscoveryEvent.h**

6.211 activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 952).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/DiscoveryEventMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.

- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.211.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 952).

NOTE!: This file is 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::generated::DiscoveryEventMarshaller::DiscoveryEventMarshaller ( ) [inline]

6.211.2.2 virtual activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::∼DiscoveryEventMarshaller ( ) [inline, virtual]

6.211.3 Member Function Documentation

6.211.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.211.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.211.3.3 virtual void activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.211.3.4 virtual void activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.211.3.5 virtual int activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.211.3.6 virtual void activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.211.3.7 virtual void activemq::wireformat::openwire::marshal::generated::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**DiscoveryEventMarshaller.h**

6.212 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** ()

6.212.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

6.212.2 Constructor & Destructor Documentation

6.212.2.1 activemq::core::DispatchData::DispatchData ( ) [inline]

6.212.2.2 activemq::core::DispatchData::DispatchData (const decaf::lang::Pointer< commands::ConsumerId > & *consumer,* const decaf::lang::Pointer< commands::Message > & *message*) [inline]

6.212.3 Member Function Documentation

6.212.3.1 const decaf::lang::Pointer<commands::ConsumerId>& activemq::core::DispatchData::getConsumerId ( ) [inline]

6.212.3.2 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**

6.213 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.

6.213.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

6.213.2 Constructor & Destructor Documentation

6.213.2.1 virtual activemq::core::Dispatcher::~Dispatcher ( ) [inline, virtual]

6.213.3 Member Function Documentation

6.213.3.1 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 to a waiting consumer.

Implemented in **activemq::core::ActiveMQSession** (p. 268), and **activemq::core::ActiveMQConsumer** (p. 185).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Dispatcher.h**

6.214 decaf::lang::Double Class Reference

`#include <src/main/decaf/lang/Double.h>`

Inheritance diagram for decaf::lang::Double:

Public Member Functions

- **Double** (double value)

  *Constructs a new instance of a **Double** (p. 956) object and assigns it the given value.*
- **Double** (**const** std::string &value)

  *Constructs a new **Double** (p. 956) and attempts to convert the given string to a double value, assigning it to the new object is successful or throwing a NumberFormatException if the string is not a properly formatted double.*
- virtual ~**Double** ()
- virtual int **compareTo** (**const Double** &d) **const**

  *Compares this **Double** (p. 956) 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. 956) 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)

  *Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 956).*
- 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. 956) instance representing the specified double value.*
- static **Double valueOf** (**const** std::string &value)

  *Returns a **Double** (p. 956) instance that wraps a primtive 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. 1543) Value.*
- static **const** double **POSITIVE_INFINITY**

  Constant for Positive Infinity.
- static **const** double **NEGATIVE_INFINITY**

  Constant for Negative Infinitiy.

6.214.1 Constructor & Destructor Documentation

6.214.1.1 decaf::lang::Double::Double (double *value*)

Constructs a new instance of a **Double** (p. 956) object and assigns it the given value.

Parameters

value	The primitive type to wrap.

6.214.1.2 decaf::lang::Double::Double (const std::string & *value*)

Constructs a new **Double** (p. 956) and attempts to convert the given string to a double value, assigning it to the new object is successful or throwing a NumberFormatException if the string is not a properly formatted double.

Parameters

value	The string to convert to a primitive type to wrap.

Exceptions

NumberFormatException	if the string is not a a valid double.

6.214.1.3 virtual decaf::lang::Double::~Double ( ) [inline, virtual]

6.214.2 Member Function Documentation

6.214.2.1 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.

Reimplemented from **decaf::lang::Number** (p. 1544).

6.214.2.2 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.

6.214.2.3 virtual int decaf::lang::Double::compareTo (const Double & *d*) const [virtual]

Compares this **Double** (p. 956) instance with another.

Parameters

d	- the **Double** (p. 956) 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 repesents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Double** > (p. 687).

6.214.2.4 virtual int decaf::lang::Double::compareTo (const double & *d*) const [virtual]

Compares this **Double** (p. 956) instance with another.

Parameters

d	- the **Double** (p. 956) 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. 687).

6.214.2.5 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

6.214.2.6 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

6.214.2.7 virtual double decaf::lang::Double::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.214.2.8 bool decaf::lang::Double::equals (const Double & *d*) const [inline, virtual]

Parameters

d	- the **Double** (p. 956) object to compare against.

Returns

true if the two **Double** (p. 956) Objects have the same value.

Implements **decaf::lang::Comparable**< **Double** > (p. 688).

6.214.2.9 bool decaf::lang::Double::equals (const double & *d*) const [inline, virtual]

Parameters

d	- the **Double** (p. 956) object to compare against.

Returns

true if the two **Double** (p. 956) Objects have the same value.

Implements **decaf::lang::Comparable**< **double** > (p. 688).

6.214.2.10 virtual float decaf::lang::Double::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.214.2.11 virtual int decaf::lang::Double::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.214.2.12 bool decaf::lang::Double::isInfinite ( ) const

Returns

true if the double is equal to positive infinity.

6.214.2.13 static bool decaf::lang::Double::isInfinite (double *value*) [static]

Parameters

value	- The double to check.

Returns

true if the double is equal to infinity.

6.214.2.14 bool decaf::lang::Double::isNaN ( ) const

Returns

true if the double is equal to NaN.

6.214.2.15 static bool decaf::lang::Double::isNaN (double *value*) [static]

Parameters

value	- The double to check.

Returns

true if the double is equal to NaN.

6.214.2.16 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 0xffffffffffffffffL, 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. 960) method.

Parameters

bits	- the long long bits to convert to double

Returns

the double converted from the bits

6.214.2.17 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.

Implements **decaf::lang::Number** (p. 1545).

6.214.2.18 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.

Implements **decaf::lang::Comparable< Double >** (p. 688).

6.214.2.19 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. 688).

6.214.2.20 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.

Implements **decaf::lang::Comparable**< **Double** > (p. 688).

6.214.2.21 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. 688).

6.214.2.22 static double decaf::lang::Double::parseDouble (const std::string *value*) [static]

Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 956).

Parameters

value	- The string to parse to an double

Returns

a double parsed from the passed string

Exceptions

NumberFormatException	

6.214.2.23 virtual short decaf::lang::Double::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1545).

6.214.2.24 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. 1173) 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.

6.214.2.25 std::string decaf::lang::Double::toString ( ) const

Returns

this **Double** (p. 956) Object as a **String** (p. 2031) Representation

6.214.2.26 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 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. 1173).

Parameters

value	- The double to convert to a string

Returns

the formatted double string.

6.214.2.27 static Double decaf::lang::Double::valueOf (double *value*) [static]

Returns a **Double** (p. 956) instance representing the specified double value.

Parameters

value	- double to wrap

Returns

new **Double** (p. 956) instance wrapping the primitive value

6.214.2.28 static Double decaf::lang::Double::valueOf (const std::string & *value*) [static]

Returns a **Double** (p. 956) instance that wraps a primtive double which is parsed from the string value passed.

Parameters

value	- the string to parse

Returns

a new **Double** (p. 956) instance wrapping the double parsed from value

Exceptions

NumberFormatException	on error.

6.214.3 Field Documentation

6.214.3.1 const double decaf::lang::Double::MAX_VALUE [static]

The maximum value that the primitive type can hold.

6.214.3.2 const double decaf::lang::Double::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.214.3.3 const double decaf::lang::Double::NaN [static]

Constant for the Not a **Number** (p. 1543) Value.

6.214.3.4 const double decaf::lang::Double::NEGATIVE_INFINITY [static]

Constant for Negative Infinitiy.

6.214.3.5 const double decaf::lang::Double::POSITIVE_INFINITY [static]

Constant for Positive Infinity.

6.214.3.6 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**

6.215 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** (int **capacity**, bool readOnly=false)

  *Creates a **DoubleArrayBuffer** (p. 966) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **DoubleArrayBuffer** (double ∗**array**, int size, int offset, int length, bool readOnly=false)

  *Creates a **DoubleArrayBuffer** (p. 966) object that wraps the given array.*
- **DoubleArrayBuffer** (**const decaf::lang::Pointer**< **ByteArrayAdapter** > &**array**, int offset, int length, bool readOnly=false)

  Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **DoubleArrayBuffer** (**const DoubleArrayBuffer** &other)

  *Create a **DoubleArrayBuffer** (p. 966) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ∼**DoubleArrayBuffer** ()
- virtual double ∗ **array** ()

  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. 451).*

Exceptions

ReadOnlyBufferException (p. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual int **arrayOffset** ()

  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. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual **DoubleBuffer** * **asReadOnlyBuffer** () **const**

  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.

- virtual **DoubleBuffer** & **compact** ()

  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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 975).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **DoubleBuffer** * **duplicate** ()

  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. 451) which the caller owns.*

- virtual double **get** ()

  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. 474)	*if there no more data to return.*

- virtual double **get** (int index) **const**

Absolute get method.
Reads the value at the given index.

Parameters

index	*The index in the **Buffer** (p. 451) 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*

- virtual bool **hasArray** () **const**

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. Subclasses 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

- virtual bool **isReadOnly** () **const**

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **DoubleBuffer** & **put** (double value)

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. 472)	*if this buffer's current position is not smaller than its limit.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **DoubleBuffer** & **put** (int index, double value)

Writes the given doubles into this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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, or the index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **DoubleBuffer** ∗ **slice** () **const**

*Creates a new **DoubleBuffer** (p. 975) 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. 975) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **DoubleArrayBuffer** (p. 966) as Read-Only or not Read-Only.*

6.215.1 Constructor & Destructor Documentation

6.215.1.1 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (int *capacity,* bool *readOnly =* `false`)

Creates a **DoubleArrayBuffer** (p. 966) 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

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArguementException	if the capacity value is negative.

6.215.1.2 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (double * *array,* int *size,* int *offset,* int *length,* bool *readOnly =* `false`)

Creates a **DoubleArrayBuffer** (p. 966) 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	The actual array to wrap.
size	The size of the given array.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if offset is greater than array capacity.

6.215.1.3 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array,* int *offset,* int *length,* bool *readOnly* = false)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **DoubleArrayBuffer** (p. 966) will be that of the remaining capacity of the passed buffer.

Parameters

array	The ByteArrayAdapter to wrap.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if array is NULL
IndexOutOfBounds-Exception	if offset + length is greater than array size.

6.215.1.4 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const DoubleArrayBuffer & *other*)

Create a **DoubleArrayBuffer** (p. 966) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.

Parameters

other	The **DoubleArrayBuffer** (p. 966) this one is to mirror.

6.215.1.5 virtual decaf::internal::nio::DoubleArrayBuffer::~DoubleArrayBuffer ( ) [virtual]

6.215.2 Member Function Documentation

6.215.2.1 virtual double* decaf::internal::nio::DoubleArrayBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 977).

6.215.2.2 virtual int decaf::internal::nio::DoubleArrayBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 977).

6.215.2.3 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. 978).

6.215.2.4 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 975).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 978).

6.215.2.5 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** (p. 451) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 978).

6.215.2.6 virtual double decaf::internal::nio::DoubleArrayBuffer::get ( ) [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. 474)*	if there no more data to return.

Implements **decaf::nio::DoubleBuffer** (p. 979).

6.215.2.7 virtual double decaf::internal::nio::DoubleArrayBuffer::get (int *index*) const [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the double is to be read.

Returns

the double that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit

Implements **decaf::nio::DoubleBuffer** (p. 979).

6.215.2.8 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. Subclasses 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. 980).

6.215.2.9 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.

Implements **decaf::nio::Buffer** (p. 454).

6.215.2.10 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (double *value*) [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. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 982).

6.215.2.11 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (int *index*, double *value*) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 982).

6.215.2.12 virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **DoubleArrayBuffer** (p. 966) as Read-Only or not Read-Only.

Parameters

value	Boolean value, true if this buffer is to be read-only, false otherwise.

6.215.2.13 virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice ( ) const [virtual]

Creates a new **DoubleBuffer** (p. 975) 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. 975) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 983).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**DoubleArrayBuffer.h**

6.216 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

Returns the double array that backs this buffer (optional operation).

- virtual int **arrayOffset** ()=0

  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

  Compacts this buffer.

- virtual **DoubleBuffer** * **duplicate** ()=0

  Creates a new double buffer that shares this buffer's content.

- virtual double **get** ()=0

  Relative get method.

- virtual double **get** (int index) **const** =0

  Absolute get method.

- **DoubleBuffer** & **get** (std::vector< double > buffer)

  Relative bulk get method.

- **DoubleBuffer** & **get** (double *buffer, int size, int offset, int length)

  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)

  This method transfers the doubles remaining in the given source buffer into this buffer.

- **DoubleBuffer** & **put** (**const** double *buffer, int size, int offset, int length)

  This method transfers doubles into this buffer from the given source array.

- **DoubleBuffer** & **put** (std::vector< double > &buffer)

  This method transfers the entire content of the given source doubles array into this buffer.

- virtual **DoubleBuffer** & **put** (double value)=0

  Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual **DoubleBuffer** & **put** (int index, double value)=0

  Writes the given doubles into this buffer at the given index.

- virtual **DoubleBuffer** * **slice** () **const** =0

  *Creates a new **DoubleBuffer** (p. 975) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (**const DoubleBuffer** &value) **const**

- virtual bool **equals** (**const DoubleBuffer** &value) **const**

- virtual bool **operator==** (**const DoubleBuffer** &value) **const**

- virtual bool **operator<** (**const DoubleBuffer** &value) **const**

Static Public Member Functions

- static **DoubleBuffer** * **allocate** (int **capacity**)

  *Allocates a new **DoubleBuffer** (p. 975).*

- static **DoubleBuffer** * **wrap** (double ***array**, int size, int offset, int length)

  *Wraps the passed buffer with a new **DoubleBuffer** (p. 975).*

- static **DoubleBuffer** * **wrap** (std::vector< double > &buffer)

  *Wraps the passed STL double Vector in a **DoubleBuffer** (p. 975).*

Protected Member Functions

- **DoubleBuffer** (int **capacity**)

  *Creates a **DoubleBuffer** (p. 975) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.216.1 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.

Since

1.0

6.216.2 Constructor & Destructor Documentation

6.216.2.1 decaf::nio::DoubleBuffer::DoubleBuffer (int *capacity*) [protected]

Creates a **DoubleBuffer** (p. 975) 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	The size and limit of the **Buffer** (p. 451) in doubles

Exceptions

IllegalArguementException	if capacity is negative.

6.216.2.2 virtual decaf::nio::DoubleBuffer::~DoubleBuffer ( ) [inline, virtual]

6.216.3 Member Function Documentation

6.216.3.1 static DoubleBuffer* decaf::nio::DoubleBuffer::allocate (int *capacity*) [static]

Allocates a new **DoubleBuffer** (p. 975).

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. 975) that was allocated, caller owns.

Exceptions

IllegalArgumentException	is the capacity value is negative.

6.216.3.2 virtual double* decaf::nio::DoubleBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 970).

6.216.3.3 virtual int decaf::nio::DoubleBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 971).

6.216.3.4 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. 971).

6.216.3.5 virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 975).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 971).

6.216.3.6 virtual int decaf::nio::DoubleBuffer::compareTo (const DoubleBuffer & *value*) const [virtual]

6.216.3.7 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. 451) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 972).

6.216.3.8 virtual bool decaf::nio::DoubleBuffer::equals (const DoubleBuffer & *value*) const [virtual]

6.216.3.9 virtual double decaf::nio::DoubleBuffer::get ( ) [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. 474)*	if there no more data to return.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 972).

6.216.3.10 virtual double decaf::nio::DoubleBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the double is to be read.

Returns

the double that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 972).

6.216.3.11 DoubleBuffer& decaf::nio::DoubleBuffer::get (std::vector< double > *buffer*)

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. 451).

Exceptions

BufferUnderflowException *(p. 474)*	iIf there are fewer than length doubles remaining in this buffer

6.216.3.12 DoubleBuffer& decaf::nio::DoubleBuffer::get (double * *buffer*, int *size*, int *offset*, int *length*)

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. 456), then no bytes are transferred and a **BufferUnderflowException** (p. 474) 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	The pointer to an allocated buffer to fill.
size	The size of the buffer passed.
offset	The position in the buffer to start filling.
length	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length doubles remaining in this buffer
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.216.3.13 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. Subclasses 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. 973).

6.216.3.14 virtual bool decaf::nio::DoubleBuffer::operator< (const DoubleBuffer & *value*) const [virtual]

6.216.3.15 virtual bool decaf::nio::DoubleBuffer::operator== (const DoubleBuffer & *value*) const [virtual]

6.216.3.16 DoubleBuffer& decaf::nio::DoubleBuffer::put (DoubleBuffer & *src*)

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. 456), then no doubles are transferred and a **BufferOverflowException** (p. 472) 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. 472)*	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. 1739)*	if this buffer is read-only.

6.216.3.17 DoubleBuffer& decaf::nio::DoubleBuffer::put (const double * *buffer*, int *size*, int *offset*, int *length*)

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. 456), then no doubles are transferred and a **BufferOverflowException** (p. 472) 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.
size	The size of the buffer passed.
offset	The offset within the array of the first char to be read.
length	The number of doubles to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.216.3.18 DoubleBuffer& decaf::nio::DoubleBuffer::put (std::vector< double > & *buffer*)

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. 975).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer.

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

6.216.3.19 virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (double *value*) [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. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 973).

6.216.3.20 virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (int *index*, double *value*) [pure virtual]

Writes the given doubles into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 974).

6.216.3.21 virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice ( ) const [pure virtual]

Creates a new **DoubleBuffer** (p. 975) 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. 975) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 974).

6.216.3.22 virtual std::string decaf::nio::DoubleBuffer::toString ( ) const [virtual]

Returns

a std::string describing this object

6.216.3.23 static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (double * *array,* int *size,* int *offset,* int *length*) [static]

Wraps the passed buffer with a new **DoubleBuffer** (p. 975).

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.
size	The size of the passed in array.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new **DoubleBuffer** (p. 975) that is backed by buffer, caller owns.

Exceptions

NullPointerException	if the array pointer is NULL.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.216.3.24 static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (std::vector< double > & *buffer*) [static]

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 975).

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. 975) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**DoubleBuffer.h**

6.217 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**

6.218 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

- **DynamicDestinationResolver** ()
- virtual ∼**DynamicDestinationResolver** () throw ()
- 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)

  Resolves the given name to a destination.

6.218.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.218.2 Constructor & Destructor Documentation

6.218.2.1 activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver ( )

6.218.2.2 virtual activemq::cmsutil::DynamicDestinationResolver::~DynamicDestinationResolver ( ) throw () [virtual]

6.218.3 Member Function Documentation

6.218.3.1 virtual void activemq::cmsutil::DynamicDestinationResolver::destroy ( ) [virtual]

Destroys any allocated resources.

Implements **activemq::cmsutil::DestinationResolver** (p. 946).

6.218.3.2 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. 985)).

Parameters

mgr	the resource lifecycle manager.

Implements **activemq::cmsutil::DestinationResolver** (p. 947).

6.218.3.3 virtual cms::Destination* activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName (cms::Session * *session,* const std::string & *destName,* bool *pubSubDomain*) [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. 640)*	if resolution failed.

Implements **activemq::cmsutil::DestinationResolver** (p. 947).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DynamicDestinationResolver.h**

6.219 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

6.219.1 Constructor & Destructor Documentation

6.219.1.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry ( ) [inline]

6.219.1.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~Entry ( ) [inline, virtual]

6.219.2 Member Function Documentation

6.219.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey ( ) const [pure virtual]

6.219.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue ( ) const [pure virtual]

6.219.2.3 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**

6.220 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 ()

6.220.1 Constructor & Destructor Documentation

6.220.1.1 decaf::io::EOFException::EOFException ( ) throw () [inline]

Default Constructor.

6.220.1.2 decaf::io::EOFException::EOFException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.220.1.3 decaf::io::EOFException::EOFException (const EOFException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.220.1.4 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

6.220.1.5 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.

6.220.1.6 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

6.220.1.7 virtual decaf::io::EOFException::~EOFException ( ) throw () [inline, virtual]

6.220.2 Member Function Documentation

6.220.2.1 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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**EOFException.h**

6.221 decaf::util::logging::ErrorManager Class Reference

ErrorManager (p. 988) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1085) during Logging.

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

Public Member Functions

- **ErrorManager** ()
- virtual ~**ErrorManager** ()
- virtual void **error** (**const** std::string &message, **decaf::lang::Exception** *ex, int **code**)

  *The error method is called when a **Handler** (p. 1085) failure occurs.*

Static Public Attributes

- static **const** int **GENERIC_FAILURE**

  GENERIC_FAILURE is used for failure that don't fit into one of the other categories.
- static **const** int **WRITE_FAILURE**

  WRITE_FAILURE is used when a write to an output stream fails.
- static **const** int **FLUSH_FAILURE**

FLUSH_FAILURE is used when a flush to an output stream fails.

- static **const** int **CLOSE_FAILURE**

  CLOSE_FAILURE is used when a close of an output stream fails.

- static **const** int **OPEN_FAILURE**

  OPEN_FAILURE is used when an open of an output stream fails.

- static **const** int **FORMAT_FAILURE**

  FORMAT_FAILURE is used when formatting fails for any reason.

6.221.1 Detailed Description

ErrorManager (p. 988) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1085) during Logging.

When processing logging output, if a **Handler** (p. 1085) encounters problems then rather than throwing an Exception back to the issuer of the logging call (who is unlikely to be interested) the **Handler** (p. 1085) should call its associated **ErrorManager** (p. 988).

Since

1.0

6.221.2 Constructor & Destructor Documentation

6.221.2.1 decaf::util::logging::ErrorManager::ErrorManager ( )

6.221.2.2 virtual decaf::util::logging::ErrorManager::~ErrorManager ( ) [virtual]

6.221.3 Member Function Documentation

6.221.3.1 virtual void decaf::util::logging::ErrorManager::error (const std::string & *message,* decaf::lang::Exception * *ex,* int *code*) [virtual]

The error method is called when a **Handler** (p. 1085) failure occurs.

This method may be overridden in subclasses. The default behavior in this base class is that the first call is reported to System.err, and subsequent calls are ignored.

Parameters

msg	- a descriptive string (may be empty)
ex	- an exception (may be NULL)
code	- an error code defined in **ErrorManager** (p. 988)

6.221.4 Field Documentation

6.221.4.1 const int decaf::util::logging::ErrorManager::CLOSE_FAILURE [static]

CLOSE_FAILURE is used when a close of an output stream fails.

6.221.4.2 const int decaf::util::logging::ErrorManager::FLUSH_FAILURE [static]

FLUSH_FAILURE is used when a flush to an output stream fails.

6.221.4.3 const int decaf::util::logging::ErrorManager::FORMAT_FAILURE [static]

FORMAT_FAILURE is used when formatting fails for any reason.

6.221.4.4 const int decaf::util::logging::ErrorManager::GENERIC_FAILURE [static]

GENERIC_FAILURE is used for failure that don't fit into one of the other categories.

6.221.4.5 const int decaf::util::logging::ErrorManager::OPEN_FAILURE [static]

OPEN_FAILURE is used when an open of an output stream fails.

6.221.4.6 const int decaf::util::logging::ErrorManager::WRITE_FAILURE [static]

WRITE_FAILURE is used when a write to an output stream fails.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**ErrorManager.h**

6.222 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.
- **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 &vargs)

Protected Attributes

- std::string **message**

  The cause of this exception.
- std::exception * **cause**

  *The **Exception** (p. 990) that caused this one to be thrown.*
- std::vector< std::pair < std::string, int > > **stackTrace**

  The stack trace.

6.222.1 Constructor & Destructor Documentation

6.222.1.1 decaf::lang::Exception::Exception ( ) throw ()

Default Constructor.

6.222.1.2 decaf::lang::Exception::Exception (const Exception & *ex*) throw ()

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) instance to copy.

6.222.1.3 decaf::lang::Exception::Exception (const std::exception * *cause*) throw ()

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.222.1.4 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

6.222.1.5 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

6.222.1.6 virtual decaf::lang::Exception::~Exception ( ) throw () [virtual]

6.222.2 Member Function Documentation

6.222.2.1 virtual void decaf::lang::Exception::buildMessage (const char * *format*, va_list & *vargs*) [protected, virtual]

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.222.2.2 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. 990) object

Implements **decaf::lang::Throwable** (p. 2117).

Reimplemented in **decaf::net::URISyntaxException** (p. 2216), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 1574), **decaf::lang::exceptions::InvalidStateException** (p. 1198), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1098), **decaf::lang::exceptions::InterruptedException** (p. 1187), **decaf::security::GeneralSecurityException** (p. 1081), **decaf::security::NoSuchProviderException** (p. 1541), **decaf::security::NoSuchAlgorithmException** (p. 1537), **decaf::security::SignatureException** (p. 1891), **decaf::lang::exceptions::IllegalStateException** (p. 1101), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1103), **decaf::lang::exceptions::NullPointerException** (p. 1543), **decaf::security::InvalidKeyException** (p. 1193), **decaf::lang::exceptions::IllegalArgumentException** (p. 1096), **decaf::util::concurrent::BrokenBarrierException** (p. 432), **decaf::lang::exceptions::RuntimeException** (p. 1796), **decaf::security::KeyException** (p. 1242), **decaf::security::KeyManagementException** (p. 1245), **decaf::util::concurrent::ExecutionException** (p. 1003), **decaf::util::concurrent::TimeoutException** (p. 2122), **decaf::util::NoSuchElementException** (p. 1539), **decaf::lang::exceptions::ClassCastException** (p. 632), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1108), **decaf::lang::exceptions::NumberFormatException** (p. 1547), **decaf::util::concurrent::CancellationException** (p. 582), **decaf::util::concurrent::RejectedExecutionException** (p. 1757), **decaf::net::BindException** (p. 413), **decaf::lang::exceptions::UnsupportedOperationException** (p. 2189), **decaf::net::ConnectException** (p. 724), **decaf::nio::InvalidMarkException** (p. 1195), **decaf::io::UTFDataFormatException** (p. 2230), **decaf::io::UnsupportedEncodingException** (p. 2187), **decaf::net::HttpRetryException** (p. 1092), **decaf::net::MalformedURLException** (p. 1371), **decaf::net::NoRouteToHostException** (p. 1534), **decaf::net::PortUnreachableException** (p. 1634), **decaf::net::ProtocolException** (p. 1715), **decaf::net::SocketTimeoutException** (p. 1934), **decaf::net::UnknownHostException** (p. 2183), **decaf::net::UnknownServiceException** (p. 2185), **decaf::util::zip::ZipException** (p. 2292), **decaf::io::EOFException** (p. 988), **decaf::io::InterruptedIOException** (p. 1189), **decaf::nio::ReadOnlyBufferException** (p. 1741), **decaf::util::ConcurrentModificationException** (p. 701), **decaf::util::zip::DataFormatException** (p. 835), **decaf::io::IOException** (p. 1200), **decaf::nio::BufferOverflowException** (p. 474), **decaf::nio::BufferUnderflowException** (p. 476), **decaf::net::SocketException** (p. 1916), **decaf::security::cert::CertificateExpiredException** (p. 590), **decaf::security::cert::CertificateNotYetValidException** (p. 591), **decaf::security::cert::CertificateParsingException** (p. 593), **decaf::security::cert::CertificateEncodingException** (p. 587), **decaf::security::cert::CertificateException** (p. 588), **activemq::exceptions::ActiveMQException** (p. 204), **activemq::exceptions::BrokerException** (p. 437), and **activemq::exceptions::ConnectionFailedException** (p. 745).

6.222.2.3 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. 2118).

6.222.2.4 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. 2118).

Referenced by activemq::exceptions::BrokerException::BrokerException().

6.222.2.5 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. 2119).

6.222.2.6 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. 2119).

6.222.2.7 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. 2119).

6.222.2.8 Exception& decaf::lang::Exception::operator= (const Exception & *ex*)

Assignment operator.

Parameters

ex	const reference to another **Exception** (p. 990)

6.222.2.9 virtual void decaf::lang::Exception::printStackTrace ( ) const [virtual]

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 2119).

6.222.2.10 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. 2119).

6.222.2.11 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. 2119).

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.222.2.12 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

6.222.2.13 virtual void decaf::lang::Exception::setStackTrace (const std::vector< std::pair< std::string, int > > & *trace*) [protected, virtual]

6.222.2.14 virtual const char* decaf::lang::Exception::what ( ) const throw () [inline, virtual]

Implement method from std::exception.

Returns

the const char* of **getMessage()** (p. 993).

6.222.3 Field Documentation

6.222.3.1 std::exception* decaf::lang::Exception::cause [protected]

The **Exception** (p. 990) that caused this one to be thrown.

6.222.3.2 std::string decaf::lang::Exception::message [protected]

The cause of this exception.

6.222.3.3 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**

6.223 cms::ExceptionListener Class Reference

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 996) that is registered with the **Connection** (p. 725).

`#include <src/main/cms/ExceptionListener.h>`

Public Member Functions

- virtual ~**ExceptionListener** ()
- virtual void **onException** (**const cms::CMSException** &ex)=0

  Called when an exception occurs.

6.223.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 996) that is registered with the **Connection** (p. 725).

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

6.223.2 Constructor & Destructor Documentation

6.223.2.1 virtual cms::ExceptionListener::~ExceptionListener ( ) [virtual]

6.223.3 Member Function Documentation

6.223.3.1 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**

6.224 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ExceptionResponse** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 Attributes

- **Pointer**< **BrokerError** > **exception**

6.224.1 Constructor & Destructor Documentation

6.224.1.1 activemq::commands::ExceptionResponse::ExceptionResponse ( )

6.224.1.2 virtual activemq::commands::ExceptionResponse::~ExceptionResponse ( ) [virtual]

6.224.2 Member Function Documentation

6.224.2.1 virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1782).

6.224.2.2 virtual void activemq::commands::ExceptionResponse::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::Response** (p. 1782).

6.224.2.3 virtual bool activemq::commands::ExceptionResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.224.2.4 virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 1783).

6.224.2.5 virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ( ) const [virtual]

6.224.2.6 virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ( ) [virtual]

6.224.2.7 virtual void activemq::commands::ExceptionResponse::setException (const Pointer< BrokerError > & *exception*) [virtual]

6.224.2.8 virtual std::string activemq::commands::ExceptionResponse::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.224.3 Field Documentation

6.224.3.1 Pointer<BrokerError> activemq::commands::ExceptionResponse::exception [protected]

6.224.3.2 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**

6.225 activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 999).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ExceptionResponseMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual ∼**ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.225.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 999).

NOTE!: This file is 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::generated::ExceptionResponseMarshaller::ExceptionResponseMarshaller ( ) [inline]

6.225.2.2 virtual activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::∼ExceptionResponseMarshaller ( ) [inline, virtual]

6.225.3 Member Function Documentation

6.225.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1789).

6.225.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.225.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.225.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.225.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.225.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.225.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ExceptionResponseMarshaller.h**

6.226 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 ()

6.226.1 Constructor & Destructor Documentation

6.226.1.1 decaf::util::concurrent::ExecutionException::ExecutionException ( ) throw () [inline]

Default Constructor.

6.226.1.2 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

6.226.1.3 decaf::util::concurrent::ExecutionException::ExecutionException (const ExecutionException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	- The Exception to copy in this new instance.

6.226.1.4 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.

6.226.1.5 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

6.226.1.6 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

6.226.1.7 virtual decaf::util::concurrent::ExecutionException::~ExecutionException ( ) throw () [inline, virtual]

6.226.2 Member Function Documentation

6.226.2.1 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. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutionException.h**

6.227 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang.Runnable** (p. 1792) tasks.

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

Inheritance diagram for decaf::util::concurrent::Executor:

Public Member Functions

- virtual ~**Executor** ()
- virtual void **execute** (**decaf::lang::Runnable** *command)=0

  Executes the given command at some time in the future.

6.227.1 Detailed Description

An object that executes submitted **decaf.lang.Runnable** (p. 1792) 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. 1004) 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. 1004) executor =  anExecutor;
executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...
```

However, the **Executor** (p. 1004) 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. 1004) {
public:

   void execute( Runnable* r ) {
        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. 1004) {
public:
    std::vector<Thread*gt; threads;

   void execute( Runnable* r ) {
        threads.push_back( new Thread( r ) );
        threads.rbegin()->start();
    }

}
```

The **Executor** (p. 1004) implementations provided in this package implement **decaf.util.concurrent.-ExecutorService** (p. 1008), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p. 2104) class provides an extensible thread pool implementation. The **decaf.util.concurrentExecutor** (p. **??**) class provides convenient factory methods for these **Executors** (p. 1005).

Since

1.0

6.227.2 Constructor & Destructor Documentation

6.227.2.1 virtual decaf::util::concurrent::Executor::~Executor ( ) [inline, virtual]

6.227.3 Member Function Documentation

6.227.3.1 virtual void decaf::util::concurrent::Executor::execute (decaf::lang::Runnable * *command*) [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. 1004) implementation.

Parameters

command	the runnable task

Exceptions

RejectedExecution-Exception *(p. 1755)*	if this task cannot be accepted for execution.
NullPointerException	if command is null

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2110).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Executor.h**

6.228 decaf::util::concurrent::Executors Class Reference

Implements a set of utilities for use with **Executors** (p. 1005), **ExecutorService** (p. 1008), **ThreadFactory** (p. 2102), and **Callable** (p. 578) types, as well as providing factory methods for instance of these types configured for the most common use cases.

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

Public Member Functions

- virtual ~**Executors** ()

Static Public Member Functions

- static **ThreadFactory** * **getDefaultThreadFactory** ()

  *Creates and returns a new **ThreadFactory** (p. 2102) that expresses the default behavior for ThreadFactories used in **Executor** (p. 1004) classes.*
- static **ExecutorService** * **newFixedThreadPool** (int nThreads)

  *Creates a new **ThreadPoolExecutor** (p. 2104) with a fixed number of threads to process incoming tasks.*
- static **ExecutorService** * **newFixedThreadPool** (int nThreads, **ThreadFactory** *threadFactory)

  *Creates a new **ThreadPoolExecutor** (p. 2104) with a fixed number of threads to process incoming tasks.*

Friends

- class **decaf::lang::Thread**

6.228.1 Detailed Description

Implements a set of utilities for use with **Executors** (p. 1005), **ExecutorService** (p. 1008), **ThreadFactory** (p. 2102), and **Callable** (p. 578) types, as well as providing factory methods for instance of these types configured for the most common use cases.

Since

1.0

6.228.2 Constructor & Destructor Documentation

6.228.2.1 virtual decaf::util::concurrent::Executors::~Executors ( ) [virtual]

6.228.3 Member Function Documentation

6.228.3.1 static ThreadFactory* decaf::util::concurrent::Executors::getDefaultThreadFactory ( ) [static]

Creates and returns a new **ThreadFactory** (p. 2102) that expresses the default behavior for ThreadFactories used in **Executor** (p. 1004) classes.

The default factory create a new non-daemon thread with normal priority and a name whose value is equal to pool-N-thread-M, where N is the sequence number of this factory, and M is the sequence number of the thread created by this factory.

Returns

a new instance of the default thread factory used in **Executors** (p. 1005), the caller takes ownership of the returned pointer.

6.228.3.2 static ExecutorService* decaf::util::concurrent::Executors::newFixedThreadPool (int *nThreads*) [static]

Creates a new **ThreadPoolExecutor** (p. 2104) with a fixed number of threads to process incoming tasks.

The thread pool will use an unbounded queue to store pending tasks. At any given time the maximum threads in the pool will be equal to the number given to this factory method. If a thread in the pool dies a new one will be spawned to take its place in the pool. Tasks that are submitted when all pooled threads are busy will be held until a thread is freed if the pool has allocated its assigned number of threads already.

Parameters

nThreads	The number of threads to assign as the max for the new **ExecutorService** (p. 1008).

Returns

pointer to a new **ExecutorService** (p. 1008) that is owned by the caller.

Exceptions

IllegalArgumentException	if nThreads is less than or equal to zero.

6.228.3.3 static ExecutorService* decaf::util::concurrent::Executors::newFixedThreadPool (int *nThreads*, ThreadFactory * *threadFactory*) [static]

Creates a new **ThreadPoolExecutor** (p. 2104) with a fixed number of threads to process incoming tasks.

The thread pool will use an unbounded queue to store pending tasks. At any given time the maximum threads in the pool will be equal to the number given to this factory method. If a thread in the pool dies a new one will be spawned to take its place in the pool. Tasks that are submitted when all pooled threads are busy will be held until a thread is freed if the pool has allocated its assigned number of threads already.

Parameters

nThreads	The number of threads to assign as the max for the new **ExecutorService** (p. 1008).
threadFactory	Instance of a **ThreadFactory** (p. 2102) that will be used by the **Executor** (p. 1004) to spawn new worker threads. This parameter cannot be NULL.

Returns

pointer to a new **ExecutorService** (p. 1008) that is owned by the caller.

Exceptions

NullPointerException	if threadFactory is NULL.
IllegalArgumentException	if nThreads is less than or equal to zero.

6.228.4 Friends And Related Function Documentation

6.228.4.1 friend class decaf::lang::Thread [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Executors.h**

6.229 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p. 1004) that provides methods to manage termination and methods that can produce a **Future** (p. 1075) 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** ()
- virtual bool **awaitTermination** (long long timeout, **const TimeUnit** &unit)=0

  The caller will block until the executor has completed termination meaning all tasks that where scheduled before shutdown have now completed and the executor is ready for deletion.
- virtual void **shutdown** ()=0

  *Performs an orderly shutdown of this **Executor** (p. 1004).*
- virtual **ArrayList** < **decaf::lang::Runnable** * > **shutdownNow** ()=0

  *Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 345) containing the Runnables that did not get executed, these object become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.*
- virtual bool **isShutdown** () **const** =0

  Returns whether this executor has been shutdown or not.
- virtual bool **isTerminated** () **const** =0

  Returns whether all tasks have completed after this executor was shut down.

6.229.1 Detailed Description

An **Executor** (p. 1004) that provides methods to manage termination and methods that can produce a **Future** (p. 1075) for tracking progress of one or more asynchronous tasks.

An **ExecutorService** (p. 1008) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p. 1008). The **shutdown()** (p. 1009) method will allow previously submitted tasks to execute before terminating, while the **shutdownNow()** (p. 1010) 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. 1008) should be shut down to allow reclamation of its resources.

Method submit extends base method **Executor.execute** (p. 1005)(**decaf.lang.Runnable** (p. 1792)) by creating and returning a **Future** (p. 1075) 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** (p. 1005) class provides factory methods for the executor services provided in this package.

Since

1.0

6.229.2 Constructor & Destructor Documentation

6.229.2.1 virtual decaf::util::concurrent::ExecutorService::~ExecutorService ( ) [inline, virtual]

6.229.3 Member Function Documentation

6.229.3.1 virtual bool decaf::util::concurrent::ExecutorService::awaitTermination (long long *timeout,* const TimeUnit & *unit*) [pure virtual]

The caller will block until the executor has completed termination meaning all tasks that where scheduled before shutdown have now completed and the executor is ready for deletion.

If the timeout period elapses before the executor reaches the terminated state then this method return false to indicate it has not terminated.

Parameters

timeout	The amount of time to wait before abandoning the wait for termination.
unit	The unit of time that the timeout value represents.

Returns

true if the executor terminated or false if the timeout expired.

Exceptions

InterruptedException	if this call is interrupted while awaiting termination.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2110).

6.229.3.2 virtual bool decaf::util::concurrent::ExecutorService::isShutdown ( ) const [pure virtual]

Returns whether this executor has been shutdown or not.

Returns

true if this executor has been shutdown.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2113).

6.229.3.3 virtual bool decaf::util::concurrent::ExecutorService::isTerminated ( ) const [pure virtual]

Returns whether all tasks have completed after this executor was shut down.

Returns

true if all tasks have completed after a request to shut down was made.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2113).

6.229.3.4 virtual void decaf::util::concurrent::ExecutorService::shutdown ( ) [pure virtual]

Performs an orderly shutdown of this **Executor** (p. 1004).

Previously queued tasks are allowed to complete but no new tasks are accepted for execution. Calling this method more than once has no affect on this executor.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2116).

6.229.3.5 virtual ArrayList<decaf::lang::Runnable*> decaf::util::concurrent::ExecutorService::shutdownNow ( ) [pure virtual]

Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 345) containing the Runnables that did not get executed, these object become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.

There is no guarantee that this method will halt execution of currently executing tasks.

Returns

an **ArrayList** (p. 345) containing all Runnable instance that were still waiting to be executed by this class, call now owns those pointers.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor** (p. 2116).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutorService.h**

6.230 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** (bool rebalance)

  *Indicates that the **Transport** (p. 2161) 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** (bool rebalance, **const List**< **URI** > &uris)

  *Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2161) is a composite of.*
- virtual void **removeURI** (bool rebalance, **const List**< **URI** > &uris)

  *Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2161) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2161) should result in that **Transport** (p. 2161) being disposed of.*
- virtual void **start** ()

  *Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.*
- virtual void **stop** ()

  *Stops the **Transport** (p. 2161).*
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- virtual void **oneway** (**const Pointer**< **Command** > &command)

  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command)

  Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command, unsigned int timeout)

Sends the given command to the broker and then waits for the response.

- virtual **Pointer**
  < **wireformat::WireFormat** > **getWireFormat** () **const**

  Gets the WireFormat instance that is in use by this transport.
- virtual void **setWireFormat** (**const Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP_UNUSED)
- virtual void **setTransportListener** (**TransportListener** *listener)

  Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () **const**

  Gets the observer of asynchronous events from this transport.
- virtual bool **isFaultTolerant** () **const**

  *Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () **const**

  *Is the **Transport** (p. 2161) Connected to its Broker.*
- virtual bool **isClosed** () **const**

  *Has the **Transport** (p. 2161) been shutdown and no longer usable.*
- bool **isInitialized** () **const**
- void **setInitialized** (bool value)
- virtual **Transport** * **narrow** (**const** std::type_info &typeId)

  *Narrows down a Chain of Transports to a specific **Transport** (p. 2161) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual std::string **getRemoteAddress** () **const**
- virtual void **reconnect** (**const decaf::net::URI** &uri)

  reconnect to another location
- virtual void **updateURIs** (bool rebalance, **const decaf::util::List**< **decaf::net::URI** > &uris)

  *Updates the set of URIs the **Transport** (p. 2161) can connect to.*
- virtual bool **isPending** () **const**
- virtual bool **iterate** ()

  *Performs the actual Reconnect operation for the **FailoverTransport** (p. 1010), 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.*
- 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)
- int **getStartupMaxReconnectAttempts** () **const**
- void **setStartupMaxReconnectAttempts** (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)
- bool **isTrackTransactionProducers** () **const**
- void **setTrackTransactionProducers** (bool value)
- int **getMaxCacheSize** () **const**
- void **setMaxCacheSize** (int value)
- bool **isReconnectSupported** () **const**
- void **setReconnectSupported** (bool value)
- bool **isUpdateURIsSupported** () **const**
- void **setUpdateURIsSupported** (bool value)
- void **setConnectionInterruptProcessingComplete** (**const Pointer**< **commands::ConnectionId** > &connectionId)

Protected Member Functions

- void **restoreTransport** (**const Pointer**< **Transport** > &transport)

  *Given a **Transport** (p. 2161) 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)

  *Called when this class' **TransportListener** (p. 2176) is notified of a Failure.*
- void **handleConnectionControl** (**const Pointer**< **Command** > &control)

  Called when the Broker sends a ConnectionControl command which could signal that this Client needs to reconnect in order to rebalance the connections on a Broker or the set of Known brokers has changed.

Friends

- class **FailoverTransportListener**

6.230.1 Constructor & Destructor Documentation

6.230.1.1 activemq::transport::failover::FailoverTransport::FailoverTransport ( )

6.230.1.2 virtual activemq::transport::failover::FailoverTransport::~FailoverTransport ( ) [virtual]

6.230.2 Member Function Documentation

6.230.2.1 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.

6.230.2.2 virtual void activemq::transport::failover::FailoverTransport::addURI (bool *rebalance*, 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. 2161) is a composite of.

Parameters

rebalance	Indicates if the addition should cause a forced reconnect or not.
uris	The new URI set to add to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 695).

6.230.2.3 virtual void activemq::transport::failover::FailoverTransport::close ( ) [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. 633).

6.230.2.4 long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier ( ) const

6.230.2.5 int activemq::transport::failover::FailoverTransport::getBackupPoolSize ( ) const

6.230.2.6 long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay ( ) const

6.230.2.7 int activemq::transport::failover::FailoverTransport::getMaxCacheSize ( ) const

6.230.2.8 int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts ( ) const

6.230.2.9 long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay ( ) const

6.230.2.10 long long activemq::transport::failover::FailoverTransport::getReconnectDelay ( ) const

6.230.2.11 virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress ( ) const [virtual]

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2162).

6.230.2.12 int activemq::transport::failover::FailoverTransport::getStartupMaxReconnectAttempts ( ) const

6.230.2.13 long long activemq::transport::failover::FailoverTransport::getTimeout ( ) const

6.230.2.14 virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener ( ) const [virtual]

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2163).

6.230.2.15 virtual Pointer<wireformat::WireFormat> activemq::transport::failover::FailoverTransport::getWireFormat ( ) const [virtual]

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a WireFormat info instance.

Returns

The WireFormat the object used to encode / decode commands.

Implements **activemq::transport::Transport** (p. 2163).

6.230.2.16 void activemq::transport::failover::FailoverTransport::handleConnectionControl (const Pointer< Command > & *control*) [protected]

Called when the Broker sends a ConnectionControl command which could signal that this Client needs to reconnect in order to rebalance the connections on a Broker or the set of Known brokers has changed.

Parameters

control	The ConnectionControl command sent from the Broker.

6.230.2.17 void activemq::transport::failover::FailoverTransport::handleTransportFailure (const decaf::lang::Exception & *error*) [protected]

Called when this class' **TransportListener** (p. 2176) is notified of a Failure.

Parameters

error	- The CMS Exception that was thrown.

Exceptions

Exception	if an error occurs.

6.230.2.18 bool activemq::transport::failover::FailoverTransport::isBackup ( ) const

6.230.2.19 virtual bool activemq::transport::failover::FailoverTransport::isClosed ( ) const [virtual]

Has the **Transport** (p. 2161) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2161)

Implements **activemq::transport::Transport** (p. 2163).

6.230.2.20 virtual bool activemq::transport::failover::FailoverTransport::isConnected ( ) const [virtual]

Is the **Transport** (p. 2161) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2163).

6.230.2.21 virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant ( ) const [inline, virtual]

Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2161) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2163).

6.230.2.22 bool activemq::transport::failover::FailoverTransport::isInitialized ( ) const

6.230.2.23 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. 692).

6.230.2.24 bool activemq::transport::failover::FailoverTransport::isRandomize ( ) const

6.230.2.25 bool activemq::transport::failover::FailoverTransport::isReconnectSupported ( ) const [virtual]

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.230.2.26 bool activemq::transport::failover::FailoverTransport::isTrackMessages ( ) const

6.230.2.27 bool activemq::transport::failover::FailoverTransport::isTrackTransactionProducers ( ) const

6.230.2.28 bool activemq::transport::failover::FailoverTransport::isUpdateURIsSupported ( ) const [virtual]

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.230.2.29 bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff ( ) const

6.230.2.30 virtual bool activemq::transport::failover::FailoverTransport::iterate ( ) [virtual]

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1010), 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. 2073).

6.230.2.31 virtual Transport* activemq::transport::failover::FailoverTransport::narrow (const std::type_info & *typeId*) [virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2161) 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. 2164).

6.230.2.32 virtual void activemq::transport::failover::FailoverTransport::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2164).

6.230.2.33 void activemq::transport::failover::FailoverTransport::reconnect (bool *rebalance*)

Indicates that the **Transport** (p. 2161) needs to reconnect to another URI in its list.

Parameters

rebalance	Indicates if the current connection should be broken and reconnected.

6.230.2.34 virtual void activemq::transport::failover::FailoverTransport::reconnect (const decaf::net::URI & *uri*) [virtual]

reconnect to another location

Parameters

uri	The new URI to connect this **Transport** (p. 2161) to.

Exceptions

IOException	on failure or if reconnect is not supported.

Implements **activemq::transport::Transport** (p. 2165).

6.230.2.35 virtual void activemq::transport::failover::FailoverTransport::removeURI (bool *rebalance,* const List< URI > & *uris*) [virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2161) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2161) should result in that **Transport** (p. 2161) being disposed of.

Parameters

rebalance	Indicates if the removal should cause a forced reconnect or not.
uris	The new URI set to remove to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 696).

6.230.2.36 virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2165).

6.230.2.37 virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & *command,* unsigned int *timeout*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2165).

6.230.2.38 void **activemq::transport::failover::FailoverTransport::restoreTransport** (const Pointer< **Transport** > & *transport*) [protected]

Given a **Transport** (p. 2161) 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. 2161) connected to the Broker.

Exceptions

IOException	if an errors occurs while restoring the old state.

6.230.2.39 void **activemq::transport::failover::FailoverTransport::setBackOffMultiplier** (long long *value*)

6.230.2.40 void **activemq::transport::failover::FailoverTransport::setBackup** (bool *value*)

6.230.2.41 void **activemq::transport::failover::FailoverTransport::setBackupPoolSize** (int *value*)

6.230.2.42 void **activemq::transport::failover::FailoverTransport::setConnectionInterrupt-ProcessingComplete** (const Pointer< **commands::ConnectionId** > & *connectionId*)

6.230.2.43 void **activemq::transport::failover::FailoverTransport::setInitialized** (bool *value*)

6.230.2.44 void **activemq::transport::failover::FailoverTransport::setInitialReconnectDelay** (long long *value*)

6.230.2.45 void **activemq::transport::failover::FailoverTransport::setMaxCacheSize** (int *value*)

6.230.2.46 void **activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts** (int *value*)

6.230.2.47 void **activemq::transport::failover::FailoverTransport::setMaxReconnectDelay** (long long *value*)

6.230.2.48 void **activemq::transport::failover::FailoverTransport::setRandomize** (bool *value*)

6.230.2.49 void **activemq::transport::failover::FailoverTransport::setReconnectDelay** (long long *value*)

6.230.2.50 void **activemq::transport::failover::FailoverTransport::setReconnectSupported** (bool *value*)

6.230.2.51 void **activemq::transport::failover::FailoverTransport::setStartupMaxReconnectAttempts** (int *value*)

6.230.2.52 void **activemq::transport::failover::FailoverTransport::setTimeout** (long long *value*)

6.230.2.53 void **activemq::transport::failover::FailoverTransport::setTrackMessages** (bool *value*)

6.230.2.54 void **activemq::transport::failover::FailoverTransport::setTrackTransactionProducers** (bool *value*)

6.230.2.55 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.

Implements **activemq::transport::Transport** (p. 2166).

6.230.2.56 void activemq::transport::failover::FailoverTransport::setUpdateURIsSupported (bool *value*)

6.230.2.57 void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff (bool *value*)

6.230.2.58 virtual void activemq::transport::failover::FailoverTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP_UNUSED*) [inline, virtual]

6.230.2.59 virtual void activemq::transport::failover::FailoverTransport::start ( ) [virtual]

Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.

Exceptions

IOException	if and error occurs while starting the **Transport** (p. 2161).

Implements **activemq::transport::Transport** (p. 2166).

6.230.2.60 virtual void activemq::transport::failover::FailoverTransport::stop ( ) [virtual]

Stops the **Transport** (p. 2161).

Exceptions

IOException	if an error occurs while stopping the transport.

Implements **activemq::transport::Transport** (p. 2167).

6.230.2.61 virtual void activemq::transport::failover::FailoverTransport::updateURIs (bool *rebalance,* const decaf::util::List< decaf::net::URI > & *uris*) [virtual]

Updates the set of URIs the **Transport** (p. 2161) can connect to.

If the **Transport** (p. 2161) doesn't support updating its URIs then an IOException is thrown.

Parameters

rebalance	Indicates if a forced reconnection should be performed as a result of the update.
uris	The new list of URIs that can be used for connection.

Exceptions

IOException	if an error occurs or updates aren't supported.

Implements **activemq::transport::Transport** (p. 2167).

6.230.3 Friends And Related Function Documentation

6.230.3.1 friend class FailoverTransportListener [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransport.h**

6.231 activemq::transport::failover::FailoverTransportFactory Class Reference

Creates an instance of a **FailoverTransport** (p. 1010).

`#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)

  *Creates a fully configured **Transport** (p. 2161) instance which could be a chain of filters and transports.*
- virtual **Pointer**< **Transport** > **createComposite** (**const decaf::net::URI** &location)

  *Creates a slimed down **Transport** (p. 2161) 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)

  *Creates a slimed down **Transport** (p. 2161) instance which can be used in composite transport instances.*

6.231.1 Detailed Description

Creates an instance of a **FailoverTransport** (p. 1010).

Since

3.0

6.231.2 Constructor & Destructor Documentation

6.231.2.1 virtual activemq::transport::failover::FailoverTransportFactory::~FailoverTransportFactory ( ) [inline, virtual]

6.231.3 Member Function Documentation

6.231.3.1 virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::create (const decaf::net::URI & *location*) [virtual]

Creates a fully configured **Transport** (p. 2161) 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. 2168).

6.231.3.2 virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::createComposite (const decaf::net::URI & *location*) [virtual]

Creates a slimed down **Transport** (p. 2161) 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. 2168).

6.231.3.3 virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::doCreateComposite (const decaf::net::URI & *location,* const decaf::util::Properties & *properties*) [protected, virtual]

Creates a slimed down **Transport** (p. 2161) 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. 1010) 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**

6.232 activemq::transport::failover::FailoverTransportListener Class Reference

Utility class used by the **Transport** (p. 2161) to perform the work of responding to events from the active **Transport** (p. 2161).

`#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.

6.232.1 Detailed Description

Utility class used by the **Transport** (p. 2161) to perform the work of responding to events from the active **Transport** (p. 2161).

Since

3.0

6.232.2 Constructor & Destructor Documentation

6.232.2.1 activemq::transport::failover::FailoverTransportListener::FailoverTransportListener (FailoverTransport ∗ *parent*)

6.232.2.2 virtual activemq::transport::failover::FailoverTransportListener::∼FailoverTransportListener ( ) [virtual]

6.232.3 Member Function Documentation

6.232.3.1 virtual void activemq::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. 2161) deletes the command upon receipt.

Parameters

command	the received command object.

Implements **activemq::transport::TransportListener** (p. 2177).

6.232.3.2 virtual void activemq::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. 2177).

6.232.3.3 virtual void activemq::transport::failover::FailoverTransportListener::transportInterrupted ( ) [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2178).

6.232.3.4 virtual void activemq::transport::failover::FailoverTransportListener::transportResumed ( ) [virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2178).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportListener.h**

6.233 activemq::core::FifoMessageDispatchChannel Class Reference

`#include <src/main/activemq/core/FifoMessageDispatchChannel.h>`

Inheritance diagram for activemq::core::FifoMessageDispatchChannel:

Public Member Functions

- **FifoMessageDispatchChannel** ()
- virtual ~**FifoMessageDispatchChannel** ()
- virtual void **enqueue** (**const Pointer**< **MessageDispatch** > &message)
  Add a Message to the Channel behind all pending message.
- virtual void **enqueueFirst** (**const Pointer**< **MessageDispatch** > &message)
  Add a message to the front of the Channel.
- virtual bool **isEmpty** () **const**
- virtual bool **isClosed** () **const**
- virtual bool **isRunning** () **const**
- virtual **Pointer**< **MessageDispatch** > **dequeue** (long long timeout)
  Used to get an enqueued message.
- virtual **Pointer**< **MessageDispatch** > **dequeueNoWait** ()
  Used to get an enqueued message if there is one queued right now.
- virtual **Pointer**< **MessageDispatch** > **peek** () **const**
  Peek in the Queue and return the first message in the Channel without removing it from the channel.
- virtual void **start** ()
  Starts dispatch of messages from the Channel.
- virtual void **stop** ()
  Stops dispatch of message from the Channel.
- virtual void **close** ()
  Close this channel no messages will be dispatched after this method is called.
- virtual void **clear** ()
  Clear the Channel, all pending messages are removed.
- virtual int **size** () **const**
- virtual 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.

6.233.1 Constructor & Destructor Documentation

6.233.1.1 activemq::core::FifoMessageDispatchChannel::FifoMessageDispatchChannel ( )

6.233.1.2 virtual activemq::core::FifoMessageDispatchChannel::~FifoMessageDispatchChannel ( ) [virtual]

6.233.2 Member Function Documentation

6.233.2.1 virtual void activemq::core::FifoMessageDispatchChannel::clear ( ) [virtual]

Clear the Channel, all pending messages are removed.

Implements **activemq::core::MessageDispatchChannel** (p. 1463).

6.233.2.2 virtual void activemq::core::FifoMessageDispatchChannel::close ( ) [virtual]

Close this channel no messages will be dispatched after this method is called.

Implements **activemq::core::MessageDispatchChannel** (p. 1463).

6.233.2.3 virtual Pointer<MessageDispatch> activemq::core::FifoMessageDispatchChannel::dequeue (long long *timeout*) [virtual]

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 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 consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

ActiveMQException	

Implements **activemq::core::MessageDispatchChannel** (p. 1463).

6.233.2.4 virtual Pointer<MessageDispatch> activemq::core::FifoMessageDispatchChannel::dequeueNoWait ( ) [virtual]

Used to get an enqueued message if there is one queued right now.

If there is no waiting message than this method returns Null.

Returns

a message if there is one in the queue.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.233.2.5 virtual void activemq::core::FifoMessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & *message*) [virtual]

Add a Message to the Channel behind all pending message.

Parameters

message	- The message to add to the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.233.2.6 virtual void activemq::core::FifoMessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & *message*) [virtual]

Add a message to the front of the Channel.

Parameters

message	- The Message to add to the front of the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.233.2.7 virtual bool activemq::core::FifoMessageDispatchChannel::isClosed ( ) const [inline, virtual]

Returns

has the Queue been closed.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.233.2.8 virtual bool activemq::core::FifoMessageDispatchChannel::isEmpty ( ) const [virtual]

Returns

true if there are no messages in the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.233.2.9 virtual bool activemq::core::FifoMessageDispatchChannel::isRunning ( ) const [inline, virtual]

Returns

true if the Channel currently running and will dequeue message.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.233.2.10 virtual void activemq::core::FifoMessageDispatchChannel::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. 2046).

6.233.2.11 virtual void activemq::core::FifoMessageDispatchChannel::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. 2047).

6.233.2.12 virtual void activemq::core::FifoMessageDispatchChannel::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. 2048).

6.233.2.13 virtual Pointer<MessageDispatch> activemq::core::FifoMessageDispatchChannel::peek ( ) const [virtual]

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.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.233.2.14 virtual std::vector< Pointer<MessageDispatch> > activemq::core::FifoMessageDispatchChannel-::removeAll ( ) [virtual]

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.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.233.2.15 virtual int activemq::core::FifoMessageDispatchChannel::size ( ) const [virtual]

Returns

the number of Messages currently in the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.233.2.16 virtual void activemq::core::FifoMessageDispatchChannel::start ( ) [virtual]

Starts dispatch of messages from the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.233.2.17 virtual void activemq::core::FifoMessageDispatchChannel::stop ( ) [virtual]

Stops dispatch of message from the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.233.2.18 virtual bool activemq::core::FifoMessageDispatchChannel::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. 2048).

6.233.2.19 virtual void activemq::core::FifoMessageDispatchChannel::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. 2049).

6.233.2.20 virtual void activemq::core::FifoMessageDispatchChannel::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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.233.2.21 virtual void activemq::core::FifoMessageDispatchChannel::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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.233.2.22 virtual void activemq::core::FifoMessageDispatchChannel::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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**FifoMessageDispatchChannel.h**

6.234 decaf::io::FileDescriptor Class Reference

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

`#include <src/main/decaf/io/FileDescriptor.h>`

Inheritance diagram for decaf::io::FileDescriptor:

Public Member Functions

- **FileDescriptor** ()
- virtual ~**FileDescriptor** ()
- void **sync** ()

  *Force any/all buffered data for this **FileDescriptor** (p. 1029) to be flushed to the underlying device.*
- bool **valid** ()

  Indicates whether the File Descriptor is valid.

Static Public Attributes

- static **FileDescriptor in**

  A handle to the standard input stream.
- static **FileDescriptor out**

  A handle to the standard output stream.
- static **FileDescriptor err**

  A handle to the standard error stream.

Protected Member Functions

- **FileDescriptor** (long value, bool **readonly**)

Protected Attributes

- long **descriptor**
- bool **readonly**

6.234.1 Detailed Description

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

Since

1.0

6.234.2 Constructor & Destructor Documentation

6.234.2.1 decaf::io::FileDescriptor::FileDescriptor (long *value,* bool *readonly*) [protected]

6.234.2.2 decaf::io::FileDescriptor::FileDescriptor ( )

6.234.2.3 virtual decaf::io::FileDescriptor::~FileDescriptor ( ) [virtual]

6.234.3 Member Function Documentation

6.234.3.1 void decaf::io::FileDescriptor::sync ( )

Force any/all buffered data for this **FileDescriptor** (p. 1029) to be flushed to the underlying device.

This method blocks until all data is flushed to the underlying device and is used to place the device into a known state. In the case of data that is buffered at a higher level such as a **BufferedOutputStream** (p. 461) the stream must first be flushed before this method can force the data to be sent to the output device.

6.234.3.2 bool decaf::io::FileDescriptor::valid ( )

Indicates whether the File Descriptor is valid.

Returns

true for a valid descriptor such as open socket or file, false otherwise.

6.234.4 Field Documentation

6.234.4.1 long decaf::io::FileDescriptor::descriptor [protected]

6.234.4.2 FileDescriptor decaf::io::FileDescriptor::err [static]

A handle to the standard error stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as System::err.

6.234.4.3 FileDescriptor decaf::io::FileDescriptor::in [static]

A handle to the standard input stream.

Usually, this file descriptor is not used directly, but rather via the input stream known as System::in.

6.234.4.4 FileDescriptor decaf::io::FileDescriptor::out [static]

A handle to the standard output stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as System::out.

6.234.4.5 bool decaf::io::FileDescriptor::readonly [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FileDescriptor.h**

6.235 decaf::util::logging::Filter Class Reference

A **Filter** (p. 1031) 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.

6.235.1 Detailed Description

A **Filter** (p. 1031) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

Each **Logger** (p. 1312) and each **Handler** (p. 1085) can have a filter associated with it. The **Logger** (p. 1312) or **Handler** (p. 1085) will call the isLoggable method to check if a given **LogRecord** (p. 1333) should be published. If isLoggable returns false, the **LogRecord** (p. 1333) will be discarded.

6.235.2 Constructor & Destructor Documentation

6.235.2.1 virtual decaf::util::logging::Filter::~Filter ( ) [inline, virtual]

6.235.3 Member Function Documentation

6.235.3.1 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. 1333) 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**

6.236 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p. 1032) 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. 1134).*
- virtual ∼**FilterInputStream** ()
- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual void **close** ()

  *Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.*
  The default implementation of this method does nothing.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs while closing the **InputStream** (p. 1134).*

- virtual long long **skip** (long long num)

  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.
  *The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)	*if an I/O error occurs.*
UnsupportedOperation-Exception	*if the concrete stream class does not support skipping bytes.*

- 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.
  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.
  Calling mark on a closed stream instance should have no effect.
  The default implementation of this method does nothing.

  Parameters

readLimit	*The max bytes read before marked position is invalid.*

- virtual void **reset** ()

  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. 1198) might be thrown.*
  - *If such an **IOException** (p. 1198) 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. 1198).*
  - *If an **IOException** (p. 1198) 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.*

  *The default implementation of this method throws an **IOException** (p. 1198).*

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual bool **markSupported** () **const**

  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.
  The default implementation of this method returns false.

  Returns

  true if this stream instance supports marks

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArray** (unsigned char ∗buffer, int size)
- virtual int **doReadArrayBounded** (unsigned char ∗buffer, int size, int offset, int length)
- virtual bool **isClosed** () **const**

Protected Attributes

- **InputStream** ∗ **inputStream**
- bool **own**
- volatile bool **closed**

6.236.1 Detailed Description

A **FilterInputStream** (p. 1032) 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. 1032) itself simply overrides all methods of **InputStream** (p. 1134) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p. 1032) may further override some of these methods and may also provide additional methods and fields.

6.236.2 Constructor & Destructor Documentation

6.236.2.1 decaf::io::FilterInputStream::FilterInputStream (InputStream * *inputStream,* bool *own* = `false`)

Constructor to create a wrapped **InputStream** (p. 1134).

Parameters

inputStream	The stream to wrap and filter.
own	Indicates if we own the stream object, defaults to false.

6.236.2.2 virtual decaf::io::FilterInputStream::~FilterInputStream ( ) [virtual]

6.236.3 Member Function Documentation

6.236.3.1 virtual int decaf::io::FilterInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1136).

Reimplemented in **decaf::io::PushbackInputStream** (p. 1718), **decaf::util::zip::InflaterInputStream** (p. 1131), and **decaf::io::BufferedInputStream** (p. 459).

6.236.3.2 virtual void decaf::io::FilterInputStream::close ( ) [virtual]

Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs while closing the **InputStream** (p. 1134).

Reimplemented from **decaf::io::InputStream** (p. 1136).

Reimplemented in **decaf::util::zip::InflaterInputStream** (p. 1131), and **decaf::io::BufferedInputStream** (p. 459).

6.236.3.3 virtual int decaf::io::FilterInputStream::doReadArray (unsigned char * *buffer,* int *size*) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.236.3.4 virtual int decaf::io::FilterInputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1136).

Reimplemented in **decaf::util::zip::InflaterInputStream** (p. 1132), **decaf::io::PushbackInputStream** (p. 1719), **decaf::io::BufferedInputStream** (p. 459), **decaf::util::zip::CheckedInputStream** (p. 626), and **activemq::io::-LoggingInputStream** (p. 1323).

6.236.3.5 virtual int decaf::io::FilterInputStream::doReadByte ( ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1137).

Reimplemented in **decaf::util::zip::InflaterInputStream** (p. 1132), **decaf::io::PushbackInputStream** (p. 1719), **decaf::io::BufferedInputStream** (p. 460), **decaf::util::zip::CheckedInputStream** (p. 626), and **activemq::io::-LoggingInputStream** (p. 1323).

6.236.3.6 virtual bool decaf::io::FilterInputStream::isClosed ( ) const [protected, virtual]

Returns

true if this stream has been closed.

6.236.3.7 virtual void decaf::io::FilterInputStream::mark (int *readLimit*) [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.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::InputStream** (p. 1137).

Reimplemented in **decaf::io::PushbackInputStream** (p. 1719), **decaf::util::zip::InflaterInputStream** (p. 1132), and **decaf::io::BufferedInputStream** (p. 460).

6.236.3.8 virtual bool decaf::io::FilterInputStream::markSupported ( ) const [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.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 1137).

Reimplemented in **decaf::io::PushbackInputStream** (p. 1719), **decaf::util::zip::InflaterInputStream** (p. 1132), and **decaf::io::BufferedInputStream** (p. 460).

6.236.3.9 virtual void decaf::io::FilterInputStream::reset ( ) [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. 1198) might be thrown.
- If such an **IOException** (p. 1198) 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. 1198).
- If an **IOException** (p. 1198) 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.

The default implementation of this method throws an **IOException** (p. 1198).

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1140).

Reimplemented in **decaf::io::PushbackInputStream** (p. 1719), **decaf::util::zip::InflaterInputStream** (p. 1133), and **decaf::io::BufferedInputStream** (p. 460).

6.236.3.10 virtual long long decaf::io::FilterInputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1140).

Reimplemented in **decaf::io::PushbackInputStream** (p. 1720), **decaf::util::zip::InflaterInputStream** (p. 1133), **decaf::io::BufferedInputStream** (p. 461), and **decaf::util::zip::CheckedInputStream** (p. 626).

6.236.4 Field Documentation

6.236.4.1 volatile bool decaf::io::FilterInputStream::closed [protected]

6.236.4.2 InputStream* decaf::io::FilterInputStream::inputStream [protected]

6.236.4.3 bool decaf::io::FilterInputStream::own [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterInputStream.h**

6.237 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 **flush** ()

  Flushes this stream by writing any buffered output to the underlying stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

  The default implementation of this method does nothing.
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
  The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1198)	*if an error occurs while closing.*

The default implementation of this method does nothing.

- virtual std::string **toString** () **const**

  Output a String representation of this object.
  The default version of this method just prints the Class Name.

  Returns

  a string representation of the object.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArray** (**const** unsigned char *buffer, int size)
- virtual void **doWriteArrayBounded** (**const** unsigned char *buffer, int size, int offset, int length)
- virtual bool **isClosed** () **const**

Protected Attributes

- **OutputStream** * **outputStream**
- bool **own**
- volatile bool **closed**

6.237.1 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. 1037) itself simply overrides all methods of **OutputStream** (p. 1600) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1037) 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. 1134) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

DataOutputStream (p. 860) os = new **DataOutputStream** (p. 860)(new **OutputStream()** (p. 1602), true)

6.237.2 Constructor & Destructor Documentation

6.237.2.1 decaf::io::FilterOutputStream::FilterOutputStream (OutputStream * *outputStream,* bool *own* = false)

Constructor, creates a wrapped output stream.

Parameters

outputStream	the **OutputStream** (p. 1600) to wrap
own	If true, this object will control the lifetime of the output stream that it encapsulates.

6.237.2.2 virtual decaf::io::FilterOutputStream::~FilterOutputStream ( ) [virtual]

6.237.3 Member Function Documentation

6.237.3.1 virtual void decaf::io::FilterOutputStream::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(p. 1198)*	if an error occurs while closing.

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1037) calls its flush method, and then calls the close method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 923).

6.237.3.2 virtual void decaf::io::FilterOutputStream::doWriteArray (const unsigned char * *buffer,* int *size*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 1602).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 462).

6.237.3.3 virtual void decaf::io::FilterOutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 1602).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 924), **decaf::io::DataOutputStream** (p. 862), **decaf::io::BufferedOutputStream** (p. 462), **decaf::util::zip::CheckedOutputStream** (p. 628), and **activemq-::io::LoggingOutputStream** (p. 1324).

6.237.3.4 virtual void decaf::io::FilterOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 1602).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 924), **decaf::io::DataOutputStream** (p. 862), **decaf::io::BufferedOutputStream** (p. 463), **decaf::util::zip::CheckedOutputStream** (p. 628), and **activemq-::io::LoggingOutputStream** (p. 1324).

6.237.3.5 virtual void decaf::io::FilterOutputStream::flush ( ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

The default implementation of this method does nothing.

The flush method of **FilterOutputStream** (p. 1037) calls the flush method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 463).

6.237.3.6 virtual bool decaf::io::FilterOutputStream::isClosed ( ) const [protected, virtual]

Returns

true if this stream has been closed.

6.237.3.7 virtual std::string decaf::io::FilterOutputStream::toString ( ) const [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

The toString method of **FilterOutputStream** (p. 1037) calls the toString method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 1603).

6.237.4 Field Documentation

6.237.4.1 volatile bool decaf::io::FilterOutputStream::closed [protected]

6.237.4.2 OutputStream* decaf::io::FilterOutputStream::outputStream [protected]

6.237.4.3 bool decaf::io::FilterOutputStream::own [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterOutputStream.h**

6.238 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)
- virtual ~**Float** ()
- virtual int **compareTo** (**const Float** &f) **const**

  *Compares this **Float*** (p. *1040) 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. 1040) 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)

  *Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1040).*
- 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. 1040) instance representing the specified float value.*
- static **Float valueOf** (**const** std::string &value)

  *Returns a **Float** (p. 1040) 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. *1543*) *Value.*
- static **const** float **POSITIVE_INFINITY**

  Constant for Positive Infinity.
- static **const** float **NEGATIVE_INFINITY**

  Constant for Negative Infinity.

6.238.1 Constructor & Destructor Documentation

6.238.1.1 decaf::lang::Float::Float (float *value*)

Parameters

value	- the primitive type to wrap

6.238.1.2 decaf::lang::Float::Float (double *value*)

Parameters

value	- the primitive type to wrap

6.238.1.3 decaf::lang::Float::Float (const std::string & *value*)

Parameters

value	- the string to convert to a primitive type to wrap

6.238.1.4 virtual decaf::lang::Float::~Float ( ) [inline, virtual]

6.238.2 Member Function Documentation

6.238.2.1 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.

Reimplemented from **decaf::lang::Number** (p. 1544).

6.238.2.2 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.

6.238.2.3 virtual int decaf::lang::Float::compareTo (const Float & *f*) const [virtual]

Compares this **Float** (p. 1040) instance with another.

Parameters

f	- the **Float** (p. 1040) 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. 687).

6.238.2.4 virtual int decaf::lang::Float::compareTo (const float & *f*) const [virtual]

Compares this **Float** (p. 1040) instance with another.

Parameters

f	- the **Float** (p. 1040) 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. 687).

6.238.2.5 virtual double decaf::lang::Float::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.238.2.6 bool decaf::lang::Float::equals (const Float & *f*) const [inline, virtual]

Parameters

f	- the **Float** (p. 1040) object to compare against.

Returns

true if the two **Float** (p. 1040) Objects have the same value.

Implements **decaf::lang::Comparable**< **Float** > (p. 688).

6.238.2.7 bool decaf::lang::Float::equals (const float & *f*) const [inline, virtual]

Parameters

f	- the **Float** (p. 1040) object to compare against.

Returns

true if the two **Float** (p. 1040) Objects have the same value.

Implements **decaf::lang::Comparable**< **float** > (p. 688).

6.238.2.8 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. 1045) 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

6.238.2.9 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. 1045) 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

6.238.2.10 virtual float decaf::lang::Float::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.238.2.11 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. 1044) method.

Parameters

bits	- the bits of the float encoded as a float

Returns

a new float created from the int bits.

6.238.2.12 virtual int decaf::lang::Float::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.238.2.13 bool decaf::lang::Float::isInfinite ( ) const

Returns

true if the float is equal to positive infinity.

6.238.2.14 static bool decaf::lang::Float::isInfinite (float *value*) [static]

Parameters

value	- The float to check.

Returns

true if the float is equal to infinity.

6.238.2.15 bool decaf::lang::Float::isNaN ( ) const

Returns

true if the float is equal to NaN.

6.238.2.16 static bool decaf::lang::Float::isNaN (float *value*) [static]

Parameters

value	- The float to check.

Returns

true if the float is equal to NaN.

6.238.2.17 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.

Implements **decaf::lang::Number** (p. 1545).

6.238.2.18 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. 688).

6.238.2.19 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.

Implements **decaf::lang::Comparable< float >** (p. 688).

6.238.2.20 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.

Implements **decaf::lang::Comparable< Float >** (p. 688).

6.238.2.21 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. 688).

6.238.2.22 static float decaf::lang::Float::parseFloat (const std::string & *value*) [static]

Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1040).

Parameters

value	- the string to parse

Returns

a float parsed from the string

Exceptions

NumberFormatException	

6.238.2.23 virtual short decaf::lang::Float::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1545).

6.238.2.24 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. 1173) 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.

6.238.2.25 std::string decaf::lang::Float::toString ( ) const

Returns

this **Float** (p. 1040) Object as a **String** (p. 2031) Representation

6.238.2.26 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. 1173).

Parameters

value	- The float to convert to a string

Returns

the formatted float string.

6.238.2.27 static Float decaf::lang::Float::valueOf (float *value*) [static]

Returns a **Float** (p. 1040) instance representing the specified float value.

Parameters

value	- float to wrap

Returns

new **Float** (p. 1040) instance wrapping the primitive value

6.238.2.28 static Float decaf::lang::Float::valueOf (const std::string & *value*) [static]

Returns a **Float** (p. 1040) 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. 1040) instance wrapping the float parsed from value

Exceptions

NumberFormatException	on error.

6.238.3 Field Documentation

6.238.3.1 const float decaf::lang::Float::MAX_VALUE [static]

The maximum value that the primitive type can hold.

6.238.3.2 const float decaf::lang::Float::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.238.3.3 const float decaf::lang::Float::NaN [static]

Constant for the Not a **Number** (p. 1543) Value.

6.238.3.4 const float decaf::lang::Float::NEGATIVE_INFINITY [static]

Constant for Negative Infinity.

6.238.3.5 const float decaf::lang::Float::POSITIVE_INFINITY [static]

Constant for Positive Infinity.

6.238.3.6 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**

6.239 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** (int size, bool readOnly=false)

  *Creates a **FloatArrayBuffer** (p. 1050) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **FloatArrayBuffer** (float ∗**array**, int size, int offset, int length, bool readOnly=false)

  *Creates a **FloatArrayBuffer** (p. 1050) object that wraps the given array.*
- **FloatArrayBuffer** (**const decaf::lang::Pointer**< **ByteArrayAdapter** > &**array**, int offset, int **capacity**, bool readOnly=false)

  Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **FloatArrayBuffer** (**const FloatArrayBuffer** &other)

  *Create a **FloatArrayBuffer** (p. 1050) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ∼**FloatArrayBuffer** ()
- virtual float ∗ **array** ()

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. 451).*

Exceptions

ReadOnlyBufferException (p. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual int **arrayOffset** ()

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. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual **FloatBuffer** * **asReadOnlyBuffer** () **const**

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.

- virtual **FloatBuffer** & **compact** ()

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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1058).*

Exceptions

ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **FloatBuffer** * **duplicate** ()

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. 451) which the caller owns.*

- virtual float **get** ()

  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. 474)	*if there no more data to return.*

- virtual float **get** (int index) **const**

  Absolute get method.
  Reads the value at the given index.

  Parameters

index	*The index in the **Buffer** (p. 451) 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*

- virtual bool **hasArray** () **const**

  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. Subclasses 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

- virtual bool **isReadOnly** () **const**

  Tells whether or not this buffer is read-only.

  Returns

  true if, and only if, this buffer is read-only.

- virtual **FloatBuffer** & **put** (float value)

  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. 472)	*if this buffer's current position is not smaller than its limit*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **FloatBuffer** & **put** (int index, float value)

  Writes the given floats into this buffer at the given index.

  Parameters

index	*The position in the **Buffer** (p. 451) 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, or index is negative.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **FloatBuffer** * **slice** () **const**

  *Creates a new **FloatBuffer** (p. 1058) 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. 1058) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **FloatArrayBuffer** (p. 1050) as Read-Only.*

6.239.1 Constructor & Destructor Documentation

6.239.1.1 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (int *size*, bool *readOnly* = `false`)

Creates a **FloatArrayBuffer** (p. 1050) 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

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArguementException	if the capacity value is negative.

6.239.1.2 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (float * *array*, int *size*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a **FloatArrayBuffer** (p. 1050) 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	The actual array to wrap.
size	The size of the given array.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if offset is greater than array capacity.

6.239.1.3 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array*, int *offset*, int *capacity*, bool *readOnly* = false)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **FloatArrayBuffer** (p. 1050) will be that of the remaining capacity of the passed buffer.

Parameters

array	The ByteArrayAdapter to wrap.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if array is NULL
IndexOutOfBounds-Exception	if offset + length is greater than array size.

6.239.1.4 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const FloatArrayBuffer & *other*)

Create a **FloatArrayBuffer** (p. 1050) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.

Parameters

other	The **FloatArrayBuffer** (p. 1050) this one is to mirror.

6.239.1.5 virtual decaf::internal::nio::FloatArrayBuffer::~FloatArrayBuffer ( ) [virtual]

6.239.2 Member Function Documentation

6.239.2.1 virtual float* decaf::internal::nio::FloatArrayBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1060).

6.239.2.2 virtual int decaf::internal::nio::FloatArrayBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1061).

6.239.2.3 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. 1061).

6.239.2.4 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1058).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1061).

6.239.2.5 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** (p. 451) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1062).

6.239.2.6 virtual float decaf::internal::nio::FloatArrayBuffer::get ( ) [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. 474)*	if there no more data to return.

Implements **decaf::nio::FloatBuffer** (p. 1062).

6.239.2.7 virtual float decaf::internal::nio::FloatArrayBuffer::get (int *index*) const [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the float is to be read

Returns

the float that is located at the given index

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit

Implements **decaf::nio::FloatBuffer** (p. 1062).

6.239.2.8 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. Subclasses 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. 1064).

6.239.2.9 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.

Implements **decaf::nio::Buffer** (p. 454).

6.239.2.10 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (float *value*) [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. 472)*	if this buffer's current position is not smaller than its limit
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1065).

6.239.2.11 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (int *index*, float *value*) [virtual]

Writes the given floats into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The floats to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::FloatBuffer** (p. 1066).

6.239.2.12 virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **FloatArrayBuffer** (p. 1050) as Read-Only.

Parameters

value	Boolean value, true if this buffer is to be read-only, false otherwise.

6.239.2.13 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice ( ) const [virtual]

Creates a new **FloatBuffer** (p. 1058) 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. 1058) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1066).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**FloatArrayBuffer.h**

6.240 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

  Returns the float array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0

  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

  Compacts this buffer.
- virtual **FloatBuffer** ∗ **duplicate** ()=0

  Creates a new float buffer that shares this buffer's content.
- virtual float **get** ()=0

  Relative get method.
- virtual float **get** (int index) **const** =0

  Absolute get method.
- **FloatBuffer** & **get** (std::vector< float > buffer)

  Relative bulk get method.
- **FloatBuffer** & **get** (float ∗buffer, int size, int offset, int length)

  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)

  This method transfers the floats remaining in the given source buffer into this buffer.
- **FloatBuffer** & **put** (**const** float ∗buffer, int size, int offset, int length)

  This method transfers floats into this buffer from the given source array.
- **FloatBuffer** & **put** (std::vector< float > &buffer)

  This method transfers the entire content of the given source floats array into this buffer.
- virtual **FloatBuffer** & **put** (float value)=0

  Writes the given floats into this buffer at the current position, and then increments the position.
- virtual **FloatBuffer** & **put** (int index, float value)=0

  Writes the given floats into this buffer at the given index.
- virtual **FloatBuffer** ∗ **slice** () **const** =0

  *Creates a new **FloatBuffer** (p. 1058) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (**const FloatBuffer** &value) **const**
- virtual bool **equals** (**const FloatBuffer** &value) **const**
- virtual bool **operator==** (**const FloatBuffer** &value) **const**
- virtual bool **operator<** (**const FloatBuffer** &value) **const**

Static Public Member Functions

- static **FloatBuffer** ∗ **allocate** (int **capacity**)

  Allocates a new Double buffer.
- static **FloatBuffer** ∗ **wrap** (float ∗**array**, int size, int offset, int length)

  *Wraps the passed buffer with a new **FloatBuffer** (p. 1058).*
- static **FloatBuffer** ∗ **wrap** (std::vector< float > &buffer)

  *Wraps the passed STL float Vector in a **FloatBuffer** (p. 1058).*

Protected Member Functions

- **FloatBuffer** (int **capacity**)

  *Creates a **FloatBuffer** (p. 1058) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.240.1 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.

6.240.2 Constructor & Destructor Documentation

6.240.2.1 decaf::nio::FloatBuffer::FloatBuffer (int *capacity*) [protected]

Creates a **FloatBuffer** (p. 1058) 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	The size and limit of the **Buffer** (p. 451) in floats.

Exceptions

IllegalArguementException	if capacity is negative.

6.240.2.2 virtual decaf::nio::FloatBuffer::~FloatBuffer ( ) [inline, virtual]

6.240.3 Member Function Documentation

6.240.3.1 static FloatBuffer* decaf::nio::FloatBuffer::allocate (int *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. 1058) that was allocated, caller owns.

6.240.3.2 virtual float* decaf::nio::FloatBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1054).

6.240.3.3 virtual int decaf::nio::FloatBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1055).

6.240.3.4 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. 1055).

6.240.3.5 virtual FloatBuffer& decaf::nio::FloatBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1058).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1055).

6.240.3.6 virtual int decaf::nio::FloatBuffer::compareTo (const FloatBuffer & *value*) const [virtual]

6.240.3.7 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. 451) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1056).

6.240.3.8 virtual bool decaf::nio::FloatBuffer::equals (const FloatBuffer & *value*) const [virtual]

6.240.3.9 virtual float decaf::nio::FloatBuffer::get ( ) [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. 474)*	if there no more data to return.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1056).

6.240.3.10 virtual float decaf::nio::FloatBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the float is to be read

Returns

the float that is located at the given index

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1056).

6.240.3.11 FloatBuffer& decaf::nio::FloatBuffer::get (std::vector< float > *buffer*)

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. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length floats remaining in this buffer

6.240.3.12 FloatBuffer& decaf::nio::FloatBuffer::get (float * *buffer*, int *size*, int *offset*, int *length*)

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. 456), then no bytes are transferred and a **BufferUnderflowException** (p. 474) 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	The pointer to an allocated buffer to fill.
size	The size of the passed in buffer.
offset	The position in the buffer to start filling.
length	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length floats remaining in this buffer
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.240.3.13 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. Subclasses 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. 1057).

6.240.3.14 virtual bool decaf::nio::FloatBuffer::operator< (const FloatBuffer & *value*) const [virtual]

6.240.3.15 virtual bool decaf::nio::FloatBuffer::operator== (const FloatBuffer & *value*) const [virtual]

6.240.3.16 FloatBuffer& decaf::nio::FloatBuffer::put (FloatBuffer & *src*)

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. 456), then no floats are transferred and a **BufferOverflowException** (p. 472) 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. 472)*	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. 1739)*	if this buffer is read-only.

6.240.3.17 FloatBuffer& decaf::nio::FloatBuffer::put (const float * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no floats are transferred and a **BufferOverflowException** (p. 472) 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.
size	The size of the passed in buffer.
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. 472)*	if there is insufficient space in this buffer
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.240.3.18 FloatBuffer& decaf::nio::FloatBuffer::put (std::vector< float > & *buffer*)

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. 1058)

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

6.240.3.19 virtual FloatBuffer& decaf::nio::FloatBuffer::put (float *value*) [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. 472)*	if this buffer's current position is not smaller than its limit
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1057).

6.240.3.20 virtual FloatBuffer& decaf::nio::FloatBuffer::put (int *index,* float *value*) [pure virtual]

Writes the given floats into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The floats to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1057).

6.240.3.21 virtual FloatBuffer* decaf::nio::FloatBuffer::slice ( ) const [pure virtual]

Creates a new **FloatBuffer** (p. 1058) 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. 1058) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1058).

6.240.3.22 virtual std::string decaf::nio::FloatBuffer::toString ( ) const [virtual]

Returns

a std::string describing this object

6.240.3.23 static FloatBuffer* decaf::nio::FloatBuffer::wrap (float * *array,* int *size,* int *offset,* int *length*) [static]

Wraps the passed buffer with a new **FloatBuffer** (p. 1058).

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.
size	The size of the array that was passed in.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new **FloatBuffer** (p. 1058) that is backed by buffer, caller owns.

Exceptions

NullPointerException	if the array pointer is NULL.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.240.3.24 static FloatBuffer* decaf::nio::FloatBuffer::wrap (std::vector< float > & *buffer*) [static]

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1058).

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. 1058) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**FloatBuffer.h**

6.241 decaf::io::Flushable Class Reference

A **Flushable** (p. 1067) is a destination of data that can be flushed.

`#include <src/main/decaf/io/Flushable.h>`

Inheritance diagram for decaf::io::Flushable:

Public Member Functions

- virtual ~**Flushable** ()
- virtual void **flush** ()=0

  Flushes this stream by writing any buffered output to the underlying stream.

6.241.1 Detailed Description

A **Flushable** (p. 1067) is a destination of data that can be flushed.

The flush method is invoked to write any buffered output to the underlying stream.

Since

1.0

6.241.2 Constructor & Destructor Documentation

6.241.2.1 virtual decaf::io::Flushable::~Flushable ( ) [inline, virtual]

6.241.3 Member Function Documentation

6.241.3.1 virtual void decaf::io::Flushable::flush ( ) [pure virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Implemented in **decaf::io::BufferedOutputStream** (p. 463), **decaf::io::FilterOutputStream** (p. 1039), **decaf::io::OutputStreamWriter** (p. 1608), **decaf::io::OutputStream** (p. 1602), **decaf::internal::io::StandardErrorOutputStream** (p. 1960), and **decaf::internal::io::StandardOutputStream** (p. 1963).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Flushable.h**

6.242 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **FlushCommand** * **cloneDataStructure** () **const**

  Clone this obbject 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. *877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure*** (p. *877) passed in to this one, and returns if they are equivalent.*
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ∗visitor)

  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

6.242.1 Constructor & Destructor Documentation

6.242.1.1 activemq::commands::FlushCommand::FlushCommand ( )

6.242.1.2 virtual activemq::commands::FlushCommand::∼FlushCommand ( ) [virtual]

6.242.2 Member Function Documentation

6.242.2.1 virtual FlushCommand∗ activemq::commands::FlushCommand::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.242.2.2 virtual void activemq::commands::FlushCommand::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. 382).

6.242.2.3 virtual bool activemq::commands::FlushCommand::equals (const DataStructure ∗ *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.242.2.4 virtual unsigned char activemq::commands::FlushCommand::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.242.2.5 virtual std::string activemq::commands::FlushCommand::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.242.2.6 virtual Pointer<Command> activemq::commands::FlushCommand::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.242.3 Field Documentation

6.242.3.1 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**

6.243 activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1070).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/FlushCommandMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.243.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1070).

NOTE!: This file is 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::generated::FlushCommandMarshaller::FlushCommandMarshaller ( ) [inline]

6.243.2.2 virtual activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::~FlushCommandMarshaller ( ) [inline, virtual]

6.243.3 Member Function Documentation

6.243.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.243.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.243.3.3 virtual void activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.243.3.4 virtual void activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.243.3.5 virtual int activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.243.3.6 virtual void activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.243.3.7 virtual void activemq::wireformat::openwire::marshal::generated::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**FlushCommandMarshaller.h**

6.244 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1074) 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**

  Format the message string from a log record.
- virtual std::string **getHead** (**const Handler** *handler **DECAF_UNUSED**)

  Return the header string for a set of formatted records.
- virtual std::string **getTail** (**const Handler** *handler **DECAF_UNUSED**)

  Return the tail string for a set of formatted records.

6.244.1 Detailed Description

A **Formatter** (p. 1074) provides support for formatting LogRecords.

Typically each logging **Handler** (p. 1085) will have a **Formatter** (p. 1074) associated with it. The **Formatter** (p. 1074) takes a **LogRecord** (p. 1333) and converts it to a string.

Some formatters (such as the **XMLFormatter** (p. 2287)) 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.

6.244.2 Constructor & Destructor Documentation

6.244.2.1 virtual decaf::util::logging::Formatter::~Formatter ( ) [inline, virtual]

6.244.3 Member Function Documentation

6.244.3.1 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::XMLFormatter** (p. 2288), and **decaf::util::logging::SimpleFormatter** (p. 1892).

6.244.3.2 virtual std::string decaf::util::logging::Formatter::formatMessage (const LogRecord & *record*) const [virtual]

Format the message string from a log record.

Parameters

record	The Log Record to Format

Returns

the formatted message

6.244.3.3 virtual std::string decaf::util::logging::Formatter::getHead (const Handler *handler *DECAF_UNUSED*) [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

the head string.

6.244.3.4 virtual std::string decaf::util::logging::Formatter::getTail (const Handler *handler *DECAF_UNUSED*) [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

the tail string

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Formatter.h**

6.245 decaf::util::concurrent::Future< V > Class Template Reference

A **Future** (p. 1075) represents the result of an asynchronous computation.

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

Public Member Functions

- virtual ~**Future** ()

- virtual bool **cancel** (bool mayInterruptIfRunning)=0

  Attempts to cancel execution of this task.
- virtual bool **isCancelled** () **const** =0

  Returns true if this task was canceled before it completed normally.
- virtual bool **isDone** () **const** =0

  Returns true if this task completed.
- virtual V **get** ()=0

  Waits if necessary for the computation to complete, and then retrieves its result.
- virtual V **get** (long long timeout, **const TimeUnit** &unit)=0

  Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

6.245.1 Detailed Description

template<typename V>class decaf::util::concurrent::Future< V >

A **Future** (p. 1075) 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. 1075) 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.

6.245.2 Constructor & Destructor Documentation

6.245.2.1 template<typename V > virtual decaf::util::concurrent::Future< V >::~Future ( ) [inline, virtual]

6.245.3 Member Function Documentation

6.245.3.1 template<typename V > virtual 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. 1077) will always return true. Subsequent calls to **isCancelled()** (p. 1077) 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

6.245.3.2 template<typename V > virtual V decaf::util::concurrent::Future< V >::get ( ) [pure virtual]

Waits if necessary for the computation to complete, and then retrieves its result.

Returns

the computed result.

Exceptions

CancellationException *(p. 580)*	- if the computation was canceled
ExecutionException *(p. 1002)*	- if the computation threw an exception
InterruptedException	- if the current thread was interrupted while waiting

6.245.3.3 template<typename V > virtual V decaf::util::concurrent::Future< V >::get (long long *timeout,* const TimeUnit & *unit*) [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. 580)*	- if the computation was canceled
ExecutionException *(p. 1002)*	- if the computation threw an exception
InterruptedException	- if the current thread was interrupted while waiting
TimeoutException *(p. 2120)*	- if the wait timed out

6.245.3.4 template<typename V > virtual 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

6.245.3.5 template<typename V > virtual 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**

6.246 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.

6.246.1 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.

6.246.2 Constructor & Destructor Documentation

6.246.2.1 activemq::transport::correlator::FutureResponse::FutureResponse ( ) [inline]

6.246.2.2 virtual activemq::transport::correlator::FutureResponse::~FutureResponse ( ) [inline, virtual]

6.246.3 Member Function Documentation

6.246.3.1 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

6.246.3.2 virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( ) [inline, virtual]

6.246.3.3 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

6.246.3.4 virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int *timeout*) [inline, virtual]

6.246.3.5 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**

6.247 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 decaf::lang::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 ()

6.247.1 Constructor & Destructor Documentation

6.247.1.1 decaf::security::GeneralSecurityException::GeneralSecurityException ( ) throw () [inline]

Default Constructor.

6.247.1.2 decaf::security::GeneralSecurityException::GeneralSecurityException (const decaf::lang::Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception

6.247.1.3 decaf::security::GeneralSecurityException::GeneralSecurityException (const GeneralSecurityException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.247.1.4 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

6.247.1.5 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.

6.247.1.6 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

6.247.1.7 virtual decaf::security::GeneralSecurityException::~GeneralSecurityException ( ) throw () [inline, virtual]

6.247.2 Member Function Documentation

6.247.2.1 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 from **decaf::lang::Exception** (p. 992).

Reimplemented in **decaf::security::NoSuchProviderException** (p. 1541), **decaf::security::NoSuchAlgorithmException** (p. 1537), **decaf::security::SignatureException** (p. 1891), **decaf::security::InvalidKeyException** (p. 1193), **decaf::security::KeyException** (p. 1242), **decaf::security::KeyManagementException** (p. 1245), **decaf::security::cert::CertificateExpiredException** (p. 590), **decaf::security::cert::CertificateNotYetValidException** (p. 591), **decaf::security::cert::CertificateParsingException** (p. 593), **decaf::security::cert::CertificateEncodingException** (p. 587), and **decaf::security::cert::CertificateException** (p. 588).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**GeneralSecurityException.h**

6.248 decaf::internal::util::GenericResource< T > Class Template Reference

A Generic **Resource** (p. 1777) wraps some type and will delete it when the **Resource** (p. 1777) itself is deleted.

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

Inheritance diagram for decaf::internal::util::GenericResource< T >:

Public Member Functions

- **GenericResource** (T ∗value)
- virtual ∼**GenericResource** ()
- T ∗ **getManaged** () **const**
- void **setManaged** (T ∗value)

6.248.1 Detailed Description

template<typename T>class decaf::internal::util::GenericResource< T >

A Generic **Resource** (p. 1777) wraps some type and will delete it when the **Resource** (p. 1777) itself is deleted.

Since

1.0

6.248.2 Constructor & Destructor Documentation

6.248.2.1 template<typename T> decaf::internal::util::GenericResource< T >::GenericResource (T ∗ *value*) [inline, explicit]

6.248.2.2 template<typename T> virtual decaf::internal::util::GenericResource< T >::∼GenericResource ( ) [inline, virtual]

6.248.3 Member Function Documentation

6.248.3.1 template<typename T> T∗ decaf::internal::util::GenericResource< T >::getManaged ( ) const [inline]

6.248.3.2 template<typename T> void decaf::internal::util::GenericResource< T >::setManaged (T ∗ *value*) [inline]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**GenericResource.h**

6.249 gz_header_s Struct Reference

`#include <src/main/decaf/internal/util/zip/zlib.h>`

Data Fields

- int **text**
- **uLong time**
- int **xflags**
- int **os**
- **Bytef** ∗ **extra**
- **uInt extra_len**
- **uInt extra_max**
- **Bytef** ∗ **name**
- **uInt name_max**
- **Bytef** ∗ **comment**

- **uInt comm_max**
- int **hcrc**
- int **done**

6.249.1 Field Documentation

6.249.1.1 uInt gz_header_s::comm_max

6.249.1.2 Bytef* gz_header_s::comment

6.249.1.3 int gz_header_s::done

6.249.1.4 Bytef* gz_header_s::extra

6.249.1.5 uInt gz_header_s::extra_len

6.249.1.6 uInt gz_header_s::extra_max

6.249.1.7 int gz_header_s::hcrc

6.249.1.8 Bytef* gz_header_s::name

6.249.1.9 uInt gz_header_s::name_max

6.249.1.10 int gz_header_s::os

6.249.1.11 int gz_header_s::text

6.249.1.12 uLong gz_header_s::time

6.249.1.13 int gz_header_s::xflags

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**zlib.h**

6.250 gz_state Struct Reference

`#include <src/main/decaf/internal/util/zip/gzguts.h>`

Data Fields

- int **mode**
- int **fd**
- char * **path**
- **z_off64_t pos**
- unsigned **size**
- unsigned **want**
- unsigned char * **in**
- unsigned char * **out**
- unsigned char * **next**
- unsigned **have**
- int **eof**

- **z_off64_t start**
- **z_off64_t raw**
- int **how**
- int **direct**
- int **level**
- int **strategy**
- **z_off64_t skip**
- int **seek**
- int **err**
- char * **msg**
- **z_stream strm**

6.250.1 Field Documentation

6.250.1.1 int gz_state::direct

6.250.1.2 int gz_state::eof

6.250.1.3 int gz_state::err

6.250.1.4 int gz_state::fd

6.250.1.5 unsigned gz_state::have

6.250.1.6 int gz_state::how

6.250.1.7 unsigned char* gz_state::in

6.250.1.8 int gz_state::level

6.250.1.9 int gz_state::mode

6.250.1.10 char* gz_state::msg

6.250.1.11 unsigned char* gz_state::next

6.250.1.12 unsigned char* gz_state::out

6.250.1.13 char* gz_state::path

6.250.1.14 z_off64_t gz_state::pos

6.250.1.15 z_off64_t gz_state::raw

6.250.1.16 int gz_state::seek

6.250.1.17 unsigned gz_state::size

6.250.1.18 z_off64_t gz_state::skip

6.250.1.19 z_off64_t gz_state::start

6.250.1.20 int gz_state::strategy

6.250.1.21 z_stream gz_state::strm

6.250.1.22 unsigned gz_state::want

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**gzguts.h**

6.251 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1085) object takes log messages from a **Logger** (p. 1312) and exports them.

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

Inheritance diagram for decaf::util::logging::Handler:

Public Member Functions

- **Handler** ()
- virtual ∼**Handler** ()
- virtual void **flush** ()=0

  *Flush the **Handler** (p. 1085)'s output, clears any buffers.*
- virtual void **publish** (**const LogRecord** &record)=0

  *Publish the Log Record to this **Handler** (p. 1085).*
- virtual bool **isLoggable** (**const LogRecord** &record) **const**

  *Check if this **Handler** (p. 1085) would actually log a given **LogRecord** (p. 1333).*
- virtual void **setFilter** (**Filter** ∗filter)

  *Sets the **Filter** (p. 1031) that this **Handler** (p. 1085) uses to filter Log Records.*
- virtual **Filter** ∗ **getFilter** ()

  *Gets the **Filter** (p. 1031) that this **Handler** (p. 1085) uses to filter Log Records.*
- virtual void **setLevel** (**const Level** &value)

  ***Set** (p. 1857) the log level specifying which message levels will be logged by this **Handler** (p. 1085).*
- virtual **Level getLevel** ()

  *Get the log level specifying which message levels will be logged by this **Handler** (p. 1085).*
- virtual void **setFormatter** (**Formatter** ∗formatter)

  *Sets the **Formatter** (p. 1074) used by this **Handler** (p. 1085).*
- virtual **Formatter** ∗ **getFormatter** ()

  *Gets the **Formatter** (p. 1074) used by this **Handler** (p. 1085).*
- virtual void **setErrorManager** (**ErrorManager** ∗errorManager)

  *Sets the **Formatter** (p. 1074) used by this **Handler** (p. 1085).*
- virtual **ErrorManager** ∗ **getErrorManager** ()

  *Gets the **ErrorManager** (p. 988) used by this **Handler** (p. 1085).*

Protected Member Functions

- void **reportError** (**const** std::string &message, **decaf::lang::Exception** ∗ex, int **code**)

  *Protected convenience method to report an error to this **Handler** (p. 1085)'s **ErrorManager** (p. 988).*

6.251.1 Detailed Description

A **Handler** (p. 1085) object takes log messages from a **Logger** (p. 1312) 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. 1085) can be disabled by doing a setLevel(**Level.OFF** (p. 1257)) and can be re-enabled by doing a setLevel with an appropriate level.

Handler (p. 1085) classes typically use **LogManager** (p. 1327) properties to set default values for the **Handler** (p. 1085)'s **Filter** (p. 1031), **Formatter** (p. 1074), and **Level** (p. 1253). See the specific documentation for each concrete **Handler** (p. 1085) class.

6.251.2 Constructor & Destructor Documentation

6.251.2.1 decaf::util::logging::Handler::Handler ( )

6.251.2.2 virtual decaf::util::logging::Handler::~Handler ( ) [virtual]

6.251.3 Member Function Documentation

6.251.3.1 virtual void decaf::util::logging::Handler::flush ( ) [pure virtual]

Flush the **Handler** (p. 1085)'s output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p. 2019).

6.251.3.2 virtual ErrorManager* decaf::util::logging::Handler::getErrorManager ( ) [inline, virtual]

Gets the **ErrorManager** (p. 988) used by this **Handler** (p. 1085).

Returns

ErrorManager (p. 988) derived pointer or NULL.

6.251.3.3 virtual Filter* decaf::util::logging::Handler::getFilter ( ) [inline, virtual]

Gets the **Filter** (p. 1031) that this **Handler** (p. 1085) uses to filter Log Records.

Returns

Filter (p. 1031) derived instance

6.251.3.4 virtual Formatter* decaf::util::logging::Handler::getFormatter ( ) [inline, virtual]

Gets the **Formatter** (p. 1074) used by this **Handler** (p. 1085).

Returns

Filter (p. 1031) derived instance

6.251.3.5 virtual Level decaf::util::logging::Handler::getLevel ( ) [inline, virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1085).

Returns

Level (p. 1253) enumeration value

6.251.3.6 virtual bool decaf::util::logging::Handler::isLoggable (const LogRecord & *record*) const [virtual]

Check if this **Handler** (p. 1085) would actually log a given **LogRecord** (p. 1333).

This method checks if the **LogRecord** (p. 1333) has an appropriate **Level** (p. 1253) and whether it satisfies any **Filter** (p. 1031). It also may make other **Handler** (p. 1085) specific checks that might prevent a handler from logging the **LogRecord** (p. 1333).

Parameters

record	**LogRecord** (p. 1333) to check

Reimplemented in **decaf::util::logging::StreamHandler** (p. 2019).

6.251.3.7 virtual void decaf::util::logging::Handler::publish (const LogRecord & *record*) [pure virtual]

Publish the Log Record to this **Handler** (p. 1085).

Parameters

record	The Log Record to Publish

Implemented in **decaf::util::logging::StreamHandler** (p. 2019), and **decaf::util::logging::ConsoleHandler** (p. 769).

6.251.3.8 void decaf::util::logging::Handler::reportError (const std::string & *message*, decaf::lang::Exception * *ex*, int *code*) [protected]

Protected convenience method to report an error to this **Handler** (p. 1085)'s **ErrorManager** (p. 988).

Parameters

message	- a descriptive string (may be empty)
ex	- an exception (may be NULL)
code	- an error code defined in **ErrorManager** (p. 988)

6.251.3.9 virtual void decaf::util::logging::Handler::setErrorManager (ErrorManager * *errorManager*) [virtual]

Sets the **Formatter** (p. 1074) used by this **Handler** (p. 1085).

The **ErrorManager** (p. 988)'s "error" method will be invoked if any errors occur while using this **Handler** (p. 1085).

Parameters

errorManager	**ErrorManager** (p. 988) derived instance

6.251.3.10 virtual void decaf::util::logging::Handler::setFilter (Filter * *filter*) [inline, virtual]

Sets the **Filter** (p. 1031) that this **Handler** (p. 1085) uses to filter Log Records.

For each call of publish the **Handler** (p. 1085) will call this **Filter** (p. 1031) (if it is non-null) to check if the **LogRecord** (p. 1333) should be published or discarded.

Parameters

filter	**Filter** (p. 1031) derived instance

6.251.3.11 virtual void decaf::util::logging::Handler::setFormatter (Formatter * *formatter*) [virtual]

Sets the **Formatter** (p. 1074) used by this **Handler** (p. 1085).

Some Handlers may not use Formatters, in which case the **Formatter** (p. 1074) will be remembered, but not used.

Parameters

formatter	**Filter** (p. 1031) derived instance

6.251.3.12 virtual void decaf::util::logging::Handler::setLevel (const Level & *value*) [inline, virtual]

Set (p. 1857) the log level specifying which message levels will be logged by this **Handler** (p. 1085).

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** (p. 1253) enumeration value

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Handler.h**

6.252 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 doulbe as needed.

Static Public Member Functions

- static double **parseDouble** (**const** std::string &hexString)
- static float **parseFloat** (**const** std::string &hexString)

6.252.1 Constructor & Destructor Documentation

6.252.1.1 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

6.252.1.2 virtual decaf::internal::util::HexStringParser::~HexStringParser ( ) [inline, virtual]

6.252.2 Member Function Documentation

6.252.2.1 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 doulbe as needed.

Parameters

hexString	- string to parse

Returns

the bits parsed from the string

6.252.2.2 static double decaf::internal::util::HexStringParser::parseDouble (const std::string & *hexString*) [static]

6.252.2.3 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**

6.253 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p. 1089) class maps hexidecimal 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)

  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**
- virtual std::size_t **size** () **const**

  Returns the max size of this Table.

6.253.1 Detailed Description

The **HexTable** (p. 1089) class maps hexidecimal strings to the value of an index into the table, i.e.

the class will return "FF" for the index 255 in the table.

6.253.2 Constructor & Destructor Documentation

6.253.2.1 activemq::wireformat::openwire::utils::HexTable::HexTable ( )

6.253.2.2 virtual activemq::wireformat::openwire::utils::HexTable::~HexTable ( ) [inline, virtual]

6.253.3 Member Function Documentation

6.253.3.1 virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t *index*) [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

IndexOutOfBounds-Exception	if the index exceeds the table size.

6.253.3.2 virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t *index*) const [virtual]

6.253.3.3 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 for the table.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**HexTable.h**

6.254 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 ()

6.254.1 Constructor & Destructor Documentation

6.254.1.1 decaf::net::HttpRetryException::HttpRetryException ( ) throw () [inline]

Default Constructor.

6.254.1.2 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

6.254.1.3 decaf::net::HttpRetryException::HttpRetryException (const HttpRetryException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.254.1.4 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

6.254.1.5 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.

6.254.1.6 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

6.254.1.7 virtual decaf::net::HttpRetryException::~HttpRetryException ( ) throw () [inline, virtual]

6.254.2 Member Function Documentation

6.254.2.1 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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**HttpRetryException.h**

6.255 activemq::util::IdGenerator Class Reference

`#include <src/main/activemq/util/IdGenerator.h>`

Public Member Functions

- **IdGenerator** ()
- **IdGenerator** (**const** std::string &prefix)
- virtual ~**IdGenerator** ()
- std::string **generateId** () **const**

Static Public Member Functions

- static std::string **getHostname** ()

  Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.
- static std::string **getSeedFromId** (**const** std::string &id)

  Gets the seed value from a Generated Id, the count portion is removed.
- static long long **getSequenceFromId** (**const** std::string &id)

  Gets the count value from a Generated Id, the seed portion is removed.
- static int **compare** (**const** std::string &id1, **const** std::string &id2)

  Compares two generated id values.

Friends

- class **activemq::library::ActiveMQCPP**

6.255.1 Constructor & Destructor Documentation

6.255.1.1 activemq::util::IdGenerator::IdGenerator ( )

6.255.1.2 activemq::util::IdGenerator::IdGenerator (const std::string & *prefix*)

6.255.1.3 virtual activemq::util::IdGenerator::~IdGenerator ( ) [virtual]

6.255.2 Member Function Documentation

6.255.2.1 static int activemq::util::IdGenerator::compare (const std::string & *id1*, const std::string & *id2*) [static]

Compares two generated id values.

Parameters

id1	The first id to compare, or left hand side.
id2	The second id to compare, or right hand side.

Returns

zero if ids are equal or positove if id1 $>$ id2...

6.255.2.2 std::string activemq::util::IdGenerator::generateId ( ) const

Returns

a newly generated unique id.

6.255.2.3 static std::string activemq::util::IdGenerator::getHostname ( ) [static]

Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.

Returns

the previously retrieved host name.

6.255.2.4 static std::string activemq::util::IdGenerator::getSeedFromId (const std::string & *id*) [static]

Gets the seed value from a Generated Id, the count portion is removed.

Returns

the seed portion of the Id, minus the count value.

6.255.2.5 static long long activemq::util::IdGenerator::getSequenceFromId (const std::string & *id*) [static]

Gets the count value from a Generated Id, the seed portion is removed.

Returns

the sequence count portion of the id, minus the seed value.

6.255.3 Friends And Related Function Documentation

6.255.3.1 friend class activemq::library::ActiveMQCPP [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/util/**IdGenerator.h**

6.256 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. 990).*
- **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 ()

6.256.1 Constructor & Destructor Documentation

6.256.1.1 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( ) throw () [inline]

Default Constructor.

6.256.1.2 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.256.1.3 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const IllegalArgumentException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.256.1.4 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.256.1.5 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

6.256.1.6 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

6.256.1.7 virtual decaf::lang::exceptions::IllegalArgumentException::~IllegalArgumentException ( ) throw () [inline, virtual]

6.256.2 Member Function Documentation

6.256.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalArgumentException.h**

6.257 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. 990).*

- **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 ()

6.257.1 Constructor & Destructor Documentation

6.257.1.1 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( ) throw () [inline]

Default Constructor.

6.257.1.2 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.257.1.3 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const IllegalMonitorStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.257.1.4 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

6.257.1.5 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

6.257.1.6 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.257.1.7 virtual decaf::lang::exceptions::IllegalMonitorStateException::~IllegalMonitorStateException ( ) throw () [inline, virtual]

6.257.2 Member Function Documentation

6.257.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalMonitorStateException.h**

6.258 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** ()
- **IllegalStateException** (**const IllegalStateException** &ex)
- **IllegalStateException** (**const** std::string &message)
- **IllegalStateException** (**const** std::string &message, **const** std::exception ∗cause)
- **IllegalStateException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ∼**IllegalStateException** () throw ()

6.258.1 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. 1834) is called on a non-transacted session.

Since

1.3

6.258.2 Constructor & Destructor Documentation

6.258.2.1 cms::IllegalStateException::IllegalStateException ( )

6.258.2.2 cms::IllegalStateException::IllegalStateException (const IllegalStateException & *ex*)

6.258.2.3 cms::IllegalStateException::IllegalStateException (const std::string & *message*)

6.258.2.4 cms::IllegalStateException::IllegalStateException (const std::string & *message,* const std::exception ∗ *cause*)

6.258.2.5 cms::IllegalStateException::IllegalStateException (const std::string & *message,* const std::exception ∗ *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.258.2.6 virtual cms::IllegalStateException::∼IllegalStateException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**IllegalStateException.h**

6.259 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. 990).*

- **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 occured.

- **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 ()

6.259.1 Constructor & Destructor Documentation

6.259.1.1 decaf::lang::exceptions::IllegalStateException::IllegalStateException ( ) throw () [inline]

Default Constructor.

6.259.1.2 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.259.1.3 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const IllegalStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.259.1.4 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 occured.

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

6.259.1.5 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

6.259.1.6 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.259.1.7 virtual decaf::lang::exceptions::IllegalStateException::~IllegalStateException ( ) throw () [inline, virtual]

6.259.2 Member Function Documentation

6.259.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 1195).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalStateException.h**

6.260 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. 990).*
- **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 occured.
- **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 ()

6.260.1 Constructor & Destructor Documentation

6.260.1.1 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( ) throw () [inline]

Default Constructor.

6.260.1.2 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.260.1.3 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const IllegalThreadStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.260.1.4 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 occured.

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

6.260.1.5 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

6.260.1.6 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.260.1.7 virtual decaf::lang::exceptions::IllegalThreadStateException::~IllegalThreadStateException ( ) throw () [inline, virtual]

6.260.2 Member Function Documentation

6.260.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalThreadStateException.h**

6.261 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** ()

  Closes this object and deallocates the appropriate resources.
- 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)

  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 **AsyncSignalReadErrorkTask**
- class **WriteChecker**
- class **AsyncWriteTask**

6.261.1 Constructor & Destructor Documentation

6.261.1.1 activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & *next,* const Pointer< wireformat::WireFormat > & *wireFormat*)

6.261.1.2 activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & *next,* const decaf::util::Properties & *properties,* const Pointer< wireformat::WireFormat > & *wireFormat*)

6.261.1.3 virtual activemq::transport::inactivity::InactivityMonitor::~InactivityMonitor ( ) [virtual]

6.261.2 Member Function Documentation

6.261.2.1 virtual void activemq::transport::inactivity::InactivityMonitor::close ( ) [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.

Reimplemented from **activemq::transport::TransportFilter** (p. 2170).

6.261.2.2 long long activemq::transport::inactivity::InactivityMonitor::getInitialDelayTime ( ) const

6.261.2.3 long long activemq::transport::inactivity::InactivityMonitor::getReadCheckTime ( ) const

6.261.2.4 long long activemq::transport::inactivity::InactivityMonitor::getWriteCheckTime ( ) const

6.261.2.5 bool activemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired ( ) const

6.261.2.6 virtual void activemq::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. 2173).

6.261.2.7 virtual void activemq::transport::inactivity::InactivityMonitor::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2173).

6.261.2.8 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. 2173).

6.261.2.9 void activemq::transport::inactivity::InactivityMonitor::setInitialDelayTime (long long *value*) const

6.261.2.10 void activemq::transport::inactivity::InactivityMonitor::setKeepAliveResponseRequired (bool *value*)

6.261.2.11 void activemq::transport::inactivity::InactivityMonitor::setReadCheckTime (long long *value*)

6.261.2.12 void activemq::transport::inactivity::InactivityMonitor::setWriteCheckTime (long long *value*)

6.261.3 Friends And Related Function Documentation

6.261.3.1 friend class AsyncSignalReadErrorkTask [friend]

6.261.3.2 friend class AsyncWriteTask [friend]

6.261.3.3 friend class ReadChecker [friend]

6.261.3.4 friend class WriteChecker [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**InactivityMonitor.h**

6.262 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. *990).*
- **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 ()

6.262.1 Constructor & Destructor Documentation

6.262.1.1 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( ) throw () [inline]

Default Constructor.

6.262.1.2 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.262.1.3 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const IndexOutOfBoundsException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.262.1.4 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

6.262.1.5 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

6.262.1.6 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.262.1.7 virtual decaf::lang::exceptions::IndexOutOfBoundsException::~IndexOutOfBoundsException ( ) throw () [inline, virtual]

6.262.2 Member Function Documentation

6.262.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IndexOutOfBoundsException.h**

6.263 decaf::net::Inet4Address Class Reference

`#include <src/main/decaf/net/Inet4Address.h>`

Inheritance diagram for decaf::net::Inet4Address:

Public Member Functions

- virtual ~**Inet4Address** ()
- virtual **InetAddress** * **clone** () **const**

  *Returns a newly allocated copy of this **InetAddress** (p. 1113).*
- virtual bool **isAnyLocalAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid wildcard address.*
- virtual bool **isLoopbackAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid loopback address.*
- virtual bool **isMulticastAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid Multicast address.*
- virtual bool **isLinkLocalAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid link local address.*
- virtual bool **isSiteLocalAddress** () **const**

*Check if this **InetAddress** (p. 1113) is a valid site local address.*

- virtual bool **isMCGlobal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Global scope.*

- virtual bool **isMCNodeLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Node Local scope.*

- virtual bool **isMCLinkLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Link Local scope.*

- virtual bool **isMCSiteLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Site Local scope.*

- virtual bool **isMCOrgLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Organization scope.*

Protected Member Functions

- **Inet4Address** ()
- **Inet4Address** (**const** unsigned char ∗ipAddress, int numBytes)
- **Inet4Address** (**const** std::string &**hostname**, **const** unsigned char ∗ipAddress, int numBytes)

Friends

- class **InetAddress**

6.263.1 Constructor & Destructor Documentation

6.263.1.1 decaf::net::Inet4Address::Inet4Address ( ) [protected]

6.263.1.2 decaf::net::Inet4Address::Inet4Address (const unsigned char ∗ *ipAddress,* int *numBytes*) [protected]

6.263.1.3 decaf::net::Inet4Address::Inet4Address (const std::string & *hostname,* const unsigned char ∗ *ipAddress,* int *numBytes*) [protected]

6.263.1.4 virtual decaf::net::Inet4Address::∼Inet4Address ( ) [virtual]

6.263.2 Member Function Documentation

6.263.2.1 virtual InetAddress∗ decaf::net::Inet4Address::clone ( ) const [virtual]

Returns a newly allocated copy of this **InetAddress** (p. 1113).

The caller owns the resulting copy and must delete it.

Returns

a new **InetAddress** (p. 1113) instance that is a copy of this one, caller owns.

Reimplemented from **decaf::net::InetAddress** (p. 1115).

6.263.2.2 virtual bool decaf::net::Inet4Address::isAnyLocalAddress ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is a valid wildcard address.

Returns

true if the address is a wildcard address.

Reimplemented from **decaf::net::InetAddress** (p. 1117).

6.263.2.3 virtual bool decaf::net::Inet4Address::isLinkLocalAddress ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is a valid link local address.

Returns

true if the address is a link local address.

Reimplemented from **decaf::net::InetAddress** (p. 1117).

6.263.2.4 virtual bool decaf::net::Inet4Address::isLoopbackAddress ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is a valid loopback address.

Returns

true if the address is a loopback address.

Reimplemented from **decaf::net::InetAddress** (p. 1117).

6.263.2.5 virtual bool decaf::net::Inet4Address::isMCGlobal ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Global scope.

Returns

true if the address is Multicast and has Global scope.

Reimplemented from **decaf::net::InetAddress** (p. 1117).

6.263.2.6 virtual bool decaf::net::Inet4Address::isMCLinkLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Link Local scope.

Returns

true if the address is Multicast and has Link Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1117).

6.263.2.7 virtual bool decaf::net::Inet4Address::isMCNodeLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Node Local scope.

Returns

true if the address is Multicast and has Node Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1118).

6.263.2.8 virtual bool decaf::net::Inet4Address::isMCOrgLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Organization scope.

Returns

true if the address is Multicast and has Organization scope.

Reimplemented from **decaf::net::InetAddress** (p. 1118).

6.263.2.9 virtual bool decaf::net::Inet4Address::isMCSiteLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Site Local scope.

Returns

true if the address is Multicast and has Site Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1118).

6.263.2.10 virtual bool decaf::net::Inet4Address::isMulticastAddress ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is a valid Multicast address.

Returns

true if the address is a Multicast address.

Reimplemented from **decaf::net::InetAddress** (p. 1118).

6.263.2.11 virtual bool decaf::net::Inet4Address::isSiteLocalAddress ( ) const [virtual]

Check if this **InetAddress** (p. 1113) is a valid site local address.

Returns

true if the address is a site local address.

Reimplemented from **decaf::net::InetAddress** (p. 1118).

6.263.3 Friends And Related Function Documentation

6.263.3.1 friend class InetAddress [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**Inet4Address.h**

6.264 decaf::net::Inet6Address Class Reference

`#include <src/main/decaf/net/Inet6Address.h>`

Inheritance diagram for decaf::net::Inet6Address:

Public Member Functions

- virtual ~**Inet6Address** ()
- virtual **InetAddress** * **clone** () **const**

  *Returns a newly allocated copy of this **InetAddress** (p. 1113).*

Protected Member Functions

- **Inet6Address** ()
- **Inet6Address** (**const** unsigned char *ipAddress, int numBytes)
- **Inet6Address** (**const** std::string &**hostname**, **const** unsigned char *ipAddress, int numBytes)

Friends

- class **InetAddress**

6.264.1 Constructor & Destructor Documentation

6.264.1.1 decaf::net::Inet6Address::Inet6Address ( ) [protected]

6.264.1.2 decaf::net::Inet6Address::Inet6Address (const unsigned char * *ipAddress,* int *numBytes*) [protected]

6.264.1.3 decaf::net::Inet6Address::Inet6Address (const std::string & *hostname,* const unsigned char * *ipAddress,* int *numBytes*) [protected]

6.264.1.4 virtual decaf::net::Inet6Address::~Inet6Address ( ) [virtual]

6.264.2 Member Function Documentation

6.264.2.1 virtual InetAddress* decaf::net::Inet6Address::clone ( ) const [virtual]

Returns a newly allocated copy of this **InetAddress** (p. 1113).

The caller owns the resulting copy and must delete it.

Returns

a new **InetAddress** (p. 1113) instance that is a copy of this one, caller owns.

Reimplemented from **decaf::net::InetAddress** (p. 1115).

6.264.3 Friends And Related Function Documentation

6.264.3.1 friend class InetAddress [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**Inet6Address.h**

6.265 decaf::net::InetAddress Class Reference

Represents an IP address.

`#include <src/main/decaf/net/InetAddress.h>`

Inheritance diagram for decaf::net::InetAddress:

Public Member Functions

- virtual ~**InetAddress** ()
- virtual **decaf::lang::ArrayPointer** < unsigned char > **getAddress** () **const**

  Returns the Raw IP address in Network byte order.
- virtual std::string **getHostAddress** () **const**

  Returns a textual representation of the IP Address.
- virtual std::string **getHostName** () **const**

  *Get the host name associated with this **InetAddress** (p. 1113) instance.*
- virtual std::string **toString** () **const**

  *Returns a string representation of the **InetAddress** (p. 1113) in the form 'hostname / ipaddress'.*
- virtual **InetAddress** * **clone** () **const**

  *Returns a newly allocated copy of this **InetAddress** (p. 1113).*
- virtual bool **isAnyLocalAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid wildcard address.*
- virtual bool **isLoopbackAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid loopback address.*
- virtual bool **isMulticastAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid Multicast address.*
- virtual bool **isLinkLocalAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid link local address.*
- virtual bool **isSiteLocalAddress** () **const**

  *Check if this **InetAddress** (p. 1113) is a valid site local address.*
- virtual bool **isMCGlobal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Global scope.*
- virtual bool **isMCNodeLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Node Local scope.*
- virtual bool **isMCLinkLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Link Local scope.*
- virtual bool **isMCSiteLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Site Local scope.*
- virtual bool **isMCOrgLocal** () **const**

  *Check if this **InetAddress** (p. 1113) is Multicast and has Organization scope.*

Static Public Member Functions

- static **InetAddress getByAddress** (**const** unsigned char *bytes, int numBytes)

  *Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 1113) instance.*
- static **InetAddress getByAddress** (**const** std::string &**hostname**, **const** unsigned char *bytes, int numBytes)

*Given a host name and IPAddress return a new **InetAddress** (p. 1113).*

- static **InetAddress getLocalHost** ()

  *Gets an **InetAddress** (p. 1113) that is the local host address.*

Protected Member Functions

- **InetAddress** ()
- **InetAddress** (**const** unsigned char *ipAddress, int numBytes)
- **InetAddress** (**const** std::string &**hostname**, **const** unsigned char *ipAddress, int numBytes)

Static Protected Member Functions

- static unsigned int **bytesToInt** (**const** unsigned char *bytes, int start)

  Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.
- static **InetAddress getAnyAddress** ()
- static **InetAddress getLoopbackAddress** ()

Protected Attributes

- std::string **hostname**
- bool **reached**
- **decaf::lang::ArrayPointer**
  < unsigned char > **addressBytes**

Static Protected Attributes

- static **const** unsigned char **loopbackBytes** [4]
- static **const** unsigned char **anyBytes** [4]

6.265.1 Detailed Description

Represents an IP address.

Since

1.0

6.265.2 Constructor & Destructor Documentation

6.265.2.1 decaf::net::InetAddress::InetAddress ( ) [protected]

6.265.2.2 decaf::net::InetAddress::InetAddress (const unsigned char * *ipAddress,* int *numBytes*) [protected]

6.265.2.3 decaf::net::InetAddress::InetAddress (const std::string & *hostname,* const unsigned char * *ipAddress,* int *numBytes*) [protected]

6.265.2.4 virtual decaf::net::InetAddress::~InetAddress ( ) [virtual]

6.265.3 Member Function Documentation

6.265.3.1 static unsigned int decaf::net::InetAddress::bytesToInt (const unsigned char * bytes, int start) [static, protected]

Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.

Parameters

bytes	The array of bytes to convert to an int.
start	The index in the array to treat as the high order byte.

Returns

an unsigned int that represents the address value.

6.265.3.2 virtual InetAddress* decaf::net::InetAddress::clone ( ) const [virtual]

Returns a newly allocated copy of this **InetAddress** (p. 1113).

The caller owns the resulting copy and must delete it.

Returns

a new **InetAddress** (p. 1113) instance that is a copy of this one, caller owns.

Reimplemented in **decaf::net::Inet4Address** (p. 1109), and **decaf::net::Inet6Address** (p. 1112).

6.265.3.3 virtual decaf::lang::ArrayPointer<unsigned char> decaf::net::InetAddress::getAddress ( ) const [virtual]

Returns the Raw IP address in Network byte order.

The returned address is a copy of the bytes contained in this **InetAddress** (p. 1113).

Returns

and ArrayPointer containing the raw bytes of the network address.

6.265.3.4 static InetAddress decaf::net::InetAddress::getAnyAddress ( ) [static, protected]

6.265.3.5 static InetAddress decaf::net::InetAddress::getByAddress (const unsigned char * bytes, int numBytes) [static]

Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 1113) instance.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

Returns

a copy of an **InetAddress** (p. 1113) that represents the given byte array address.

Exceptions

UnknownHostException *(p. 2181)*	if the address array length is invalid.

6.265.3.6 static InetAddress decaf::net::InetAddress::getByAddress (const std::string & *hostname,* const unsigned char * *bytes,* int *numBytes*) [static]

Given a host name and IPAddress return a new **InetAddress** (p. 1113).

There is no name service checking or address validation done on the provided host name. The host name can either be machine name or the text based representation of the IP Address.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

Returns

a copy of an **InetAddress** (p. 1113) that represents the given byte array address.

Exceptions

UnknownHostException *(p. 2181)*	if the address array length is invalid.

6.265.3.7 virtual std::string decaf::net::InetAddress::getHostAddress ( ) const [virtual]

Returns a textual representation of the IP Address.

Returns

the string form of the IP Address.

6.265.3.8 virtual std::string decaf::net::InetAddress::getHostName ( ) const [virtual]

Get the host name associated with this **InetAddress** (p. 1113) instance.

If a host name was set upon construction then that value is returned, otherwise a reverse name lookup with be performed to attempt to get the host name associated with the set IP Address. If the host name cannot be resolved the textual representation of the IP Address is returned instead.

Returns

the name of the host associated with this set IP Address.

6.265.3.9 static InetAddress decaf::net::InetAddress::getLocalHost ( ) [static]

Gets an **InetAddress** (p. 1113) that is the local host address.

If the localhost value cannot be resolved than the **InetAddress** (p. 1113) for Loopback is returned.

Returns

a new **InetAddress** (p. 1113) object that contains the local host address.

Exceptions

UnknownHostException *(p. 2181)*	if the address for local host is not found.

6.265.3.10 static InetAddress decaf::net::InetAddress::getLoopbackAddress ( ) [static, protected]

6.265.3.11 virtual bool decaf::net::InetAddress::isAnyLocalAddress ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is a valid wildcard address.

Returns

true if the address is a wildcard address.

Reimplemented in **decaf::net::Inet4Address** (p. 1109).

6.265.3.12 virtual bool decaf::net::InetAddress::isLinkLocalAddress ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is a valid link local address.

Returns

true if the address is a link local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1110).

6.265.3.13 virtual bool decaf::net::InetAddress::isLoopbackAddress ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is a valid loopback address.

Returns

true if the address is a loopback address.

Reimplemented in **decaf::net::Inet4Address** (p. 1110).

6.265.3.14 virtual bool decaf::net::InetAddress::isMCGlobal ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Global scope.

Returns

true if the address is Multicast and has Global scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1110).

6.265.3.15 virtual bool decaf::net::InetAddress::isMCLinkLocal ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Link Local scope.

Returns

true if the address is Multicast and has Link Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1110).

6.265.3.16 virtual bool decaf::net::InetAddress::isMCNodeLocal ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Node Local scope.

Returns

true if the address is Multicast and has Node Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1110).

6.265.3.17 virtual bool decaf::net::InetAddress::isMCOrgLocal ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Organization scope.

Returns

true if the address is Multicast and has Organization scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1111).

6.265.3.18 virtual bool decaf::net::InetAddress::isMCSiteLocal ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is Multicast and has Site Local scope.

Returns

true if the address is Multicast and has Site Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1111).

6.265.3.19 virtual bool decaf::net::InetAddress::isMulticastAddress ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is a valid Multicast address.

Returns

true if the address is a Multicast address.

Reimplemented in **decaf::net::Inet4Address** (p. 1111).

6.265.3.20 virtual bool decaf::net::InetAddress::isSiteLocalAddress ( ) const [inline, virtual]

Check if this **InetAddress** (p. 1113) is a valid site local address.

Returns

true if the address is a site local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1111).

6.265.3.21 virtual std::string decaf::net::InetAddress::toString ( ) const [virtual]

Returns a string representation of the **InetAddress** (p. 1113) in the form 'hostname / ipaddress'.

If the hostname is not resolved than it appears as empty.

Returns

string value of this **InetAddress** (p. 1113).

6.265.4 Field Documentation

6.265.4.1 decaf::lang::ArrayPointer<unsigned char> decaf::net::InetAddress::addressBytes [mutable, protected]

6.265.4.2 const unsigned char decaf::net::InetAddress::anyBytes[4] [static, protected]

6.265.4.3 std::string decaf::net::InetAddress::hostname [mutable, protected]

6.265.4.4 const unsigned char decaf::net::InetAddress::loopbackBytes[4] [static, protected]

6.265.4.5 bool decaf::net::InetAddress::reached [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**InetAddress.h**

6.266 decaf::net::InetSocketAddress Class Reference

`#include <src/main/decaf/net/InetSocketAddress.h>`

Inheritance diagram for decaf::net::InetSocketAddress:

Public Member Functions

- **InetSocketAddress** ()
- virtual ~**InetSocketAddress** ()

6.266.1 Constructor & Destructor Documentation

6.266.1.1 decaf::net::InetSocketAddress::InetSocketAddress ( )

6.266.1.2 virtual decaf::net::InetSocketAddress::~InetSocketAddress ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**InetSocketAddress.h**

6.267 inflate_state Struct Reference

`#include <src/main/decaf/internal/util/zip/inflate.h>`

Data Fields

- **inflate_mode mode**
- int **last**
- int **wrap**
- int **havedict**
- int **flags**
- unsigned **dmax**

- unsigned long **check**
- unsigned long **total**
- **gz_headerp head**
- unsigned **wbits**
- unsigned **wsize**
- unsigned **whave**
- unsigned **wnext**
- unsigned char **FAR** ∗ **window**
- unsigned long **hold**
- unsigned **bits**
- unsigned **length**
- unsigned **offset**
- unsigned **extra**
- **code const FAR** ∗ **lencode**
- **code const FAR** ∗ **distcode**
- unsigned **lenbits**
- unsigned **distbits**
- unsigned **ncode**
- unsigned **nlen**
- unsigned **ndist**
- unsigned **have**
- **code FAR** ∗ **next**
- unsigned short **lens** [320]
- unsigned short **work** [288]
- **code codes** [**ENOUGH**]
- int **sane**
- int **back**
- unsigned **was**

6.267.1 Field Documentation

6.267.1.1 int inflate_state::back

6.267.1.2 unsigned inflate_state::bits

6.267.1.3 unsigned long inflate_state::check

6.267.1.4 code inflate_state::codes[ENOUGH]

6.267.1.5 unsigned inflate_state::distbits

6.267.1.6 code const FAR∗ inflate_state::distcode

6.267.1.7 unsigned inflate_state::dmax

6.267.1.8 unsigned inflate_state::extra

6.267.1.9 int inflate_state::flags

6.267.1.10 unsigned inflate_state::have

6.267.1.11 int inflate_state::havedict

6.267.1.12 gz_headerp inflate_state::head

6.267.1.13 unsigned long inflate_state::hold

6.267.1.14 int inflate_state::last

6.267.1.15 unsigned inflate_state::lenbits

6.267.1.16 code const FAR* inflate_state::lencode

6.267.1.17 unsigned inflate_state::length

6.267.1.18 unsigned short inflate_state::lens[320]

6.267.1.19 inflate_mode inflate_state::mode

6.267.1.20 unsigned inflate_state::ncode

6.267.1.21 unsigned inflate_state::ndist

6.267.1.22 code FAR* inflate_state::next

6.267.1.23 unsigned inflate_state::nlen

6.267.1.24 unsigned inflate_state::offset

6.267.1.25 int inflate_state::sane

6.267.1.26 unsigned long inflate_state::total

6.267.1.27 unsigned inflate_state::was

6.267.1.28 unsigned inflate_state::wbits

6.267.1.29 unsigned inflate_state::whave

6.267.1.30 unsigned char FAR* inflate_state::window

6.267.1.31 unsigned inflate_state::wnext

6.267.1.32 unsigned short inflate_state::work[288]

6.267.1.33 int inflate_state::wrap

6.267.1.34 unsigned inflate_state::wsize

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**inflate.h**

6.268 decaf::util::zip::Inflater Class Reference

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see `specification`).

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

Public Member Functions

- **Inflater** ()

  Creates a new decompressor.
- **Inflater** (bool nowrap)

  Creates a new decompressor.
- virtual ∼**Inflater** ()
- void **setInput** (**const** unsigned char ∗buffer, int size, int offset, int length)

  Sets input data for decompression.
- void **setInput** (**const** std::vector< unsigned char > &buffer, int offset, int length)

  Sets input data for decompression.
- void **setInput** (**const** std::vector< unsigned char > &buffer)

  Sets input data for decompression.
- int **getRemaining** () **const**

  Returns the total number of bytes remaining in the input buffer.
- void **setDictionary** (**const** unsigned char ∗buffer, int size, int offset, int length)

  Sets the preset dictionary to the given array of bytes.
- void **setDictionary** (**const** std::vector< unsigned char > &buffer, int offset, int length)

  Sets the preset dictionary to the given array of bytes.
- void **setDictionary** (**const** std::vector< unsigned char > &buffer)

  Sets the preset dictionary to the given array of bytes.
- bool **needsInput** () **const**
- bool **needsDictionary** () **const**
- void **finish** ()

  When called, indicates that decompression should end with the current contents of the input buffer.
- bool **finished** () **const**
- int **inflate** (unsigned char ∗buffer, int size, int offset, int length)

  Uncompresses bytes into specified buffer.
- int **inflate** (std::vector< unsigned char > &buffer, int offset, int length)

  Uncompresses bytes into specified buffer.
- int **inflate** (std::vector< unsigned char > &buffer)

  Uncompresses bytes into specified buffer.
- long long **getAdler** () **const**
- long long **getBytesRead** () **const**
- long long **getBytesWritten** () **const**
- void **reset** ()

  Resets deflater so that a new set of input data can be processed.
- void **end** ()

  Closes the decompressor and discards any unprocessed input.

6.268.1 Detailed Description

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see `specification`).

Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **InflaterInputStream** (p. 1128) and its descendants.

The typical usage of a **Inflater** (p. 1121) outside this package consists of a specific call to one of its constructors before being passed to an instance of **InflaterInputStream** (p. 1128).

See also

InflaterInputStream (p. 1128)
Deflater (p. 913)

Since

1.0

6.268.2 Constructor & Destructor Documentation

6.268.2.1 decaf::util::zip::Inflater::Inflater ( )

Creates a new decompressor.

This constructor defaults the inflater to use the ZLIB header and checksum fields.

6.268.2.2 decaf::util::zip::Inflater::Inflater (bool *nowrap*)

Creates a new decompressor.

If the parameter 'nowrap' is true then the ZLIB header and checksum fields will not be used. This provides compatibility with the compression format used by both GZIP and PKZIP.

Note: When using the 'nowrap' option it is also necessary to provide an extra "dummy" byte as input. This is required by the ZLIB native library in order to support certain optimizations.

6.268.2.3 virtual decaf::util::zip::Inflater::~Inflater ( ) [virtual]

6.268.3 Member Function Documentation

6.268.3.1 void decaf::util::zip::Inflater::end ( )

Closes the decompressor and discards any unprocessed input.

This method should be called when the decompressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Inflater** (p. 1121) object is undefined.

6.268.3.2 void decaf::util::zip::Inflater::finish ( )

When called, indicates that decompression should end with the current contents of the input buffer.

6.268.3.3 bool decaf::util::zip::Inflater::finished ( ) const

Returns

true if the end of the compressed data output stream has been reached.

6.268.3.4 long long decaf::util::zip::Inflater::getAdler ( ) const

Returns

the ADLER-32 value of the uncompressed data.

Exceptions

IllegalStateException	if in the end state.

6.268.3.5 long long decaf::util::zip::Inflater::getBytesRead ( ) const

Returns

the total number of compressed bytes input so far.

Exceptions

IllegalStateException	if in the end state.

6.268.3.6 long long decaf::util::zip::Inflater::getBytesWritten ( ) const

Returns

the total number of decompressed bytes output so far.

Exceptions

IllegalStateException	if in the end state.

6.268.3.7 int decaf::util::zip::Inflater::getRemaining ( ) const

Returns the total number of bytes remaining in the input buffer.

This can be used to find out what bytes still remain in the input buffer after decompression has finished.

Returns

the total number of bytes remaining in the input buffer

6.268.3.8 int decaf::util::zip::Inflater::inflate (unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1125) or **needsDictionary()** (p. 1125) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1123) can be used to get the Adler-32 value of the dictionary required.

Parameters

buffer	The Buffer to write the compressed data to.
size	The size of the buffer passed in.
offset	The position in the Buffer to start writing at.
length	The maximum number of byte of data to write.

Exceptions

NullPointerException	if buffer is NULL.
IllegalStateException	if in the end state.
IndexOutOfBounds-Exception	if the offset + length $>$ size of the buffer.
DataFormatException *(p. 834)*	if the compressed data format is invalid.

6.268.3.9 int decaf::util::zip::Inflater::inflate (std::vector< unsigned char > & *buffer,* int *offset,* int *length*)

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1125) or **needsDictionary()** (p. 1125) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1123) can be used to get the Adler-32 value of the dictionary required.

Parameters

buffer	The Buffer to write the compressed data to.
offset	The position in the Buffer to start writing at.
length	The maximum number of byte of data to write.

Exceptions

IllegalStateException	if in the end state.
IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
DataFormatException *(p. 834)*	if the compressed data format is invalid.

6.268.3.10 int decaf::util::zip::Inflater::inflate (std::vector< unsigned char > & *buffer*)

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1125) or **needsDictionary()** (p. 1125) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1123) can be used to get the Adler-32 value of the dictionary required.

Parameters

buffer	The Buffer to write the compressed data to.

Exceptions

IllegalStateException	if in the end state.
DataFormatException *(p. 834)*	if the compressed data format is invalid.

6.268.3.11 bool decaf::util::zip::Inflater::needsDictionary ( ) const

Returns

true if a preset dictionary is needed for decompression.

6.268.3.12 bool decaf::util::zip::Inflater::needsInput ( ) const

Returns

true if the input data buffer is empty and **setInput()** (p. 1127) should be called in order to provide more input

6.268.3.13 void decaf::util::zip::Inflater::reset ( )

Resets deflater so that a new set of input data can be processed.

Keeps current decompression level and strategy settings.

Exceptions

IllegalStateException	if in the end state.

6.268.3.14 void decaf::util::zip::Inflater::setDictionary (const unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1124) returns 0 and **needsDictionary()** (p. 1125) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1123) can be used to get the Adler-32 value of the dictionary needed.

Parameters

buffer	The Buffer to read in for decompression.
size	The size of the buffer passed in.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

NullPointerException	if buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length $>$ size of the buffer.
IllegalStateException	if in the end state.
IllegalArgumentException	if the given dictionary doesn't match thre required dictionaries checksum value.

6.268.3.15 void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & *buffer,* int *offset,* int *length*)

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1124) returns 0 and **needsDictionary()** (p. 1125) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1123) can be used to get the Adler-32 value of the dictionary needed.

Parameters

buffer	The Buffer to read in for decompression.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBounds-Exception	if the offset + length $>$ size of the buffer.
IllegalStateException	if in the end state.
IllegalArgumentException	if the given dictionary doesn't match thre required dictionaries checksum value.

6.268.3.16 void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & *buffer*)

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1124) returns 0 and **needsDictionary()** (p. 1125) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1123) can be used to get the Adler-32 value of the dictionary needed.

Parameters

buffer	The Buffer to read in for decompression.

Exceptions

IllegalStateException	if in the end state.
IllegalArgumentException	if the given dictionary doesn't match the required dictionaries checksum value.

6.268.3.17 void decaf::util::zip::Inflater::setInput (const unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1125) returns true indicating that more input data is required.

Parameters

buffer	The Buffer to read in for decompression.
size	The size of the buffer passed in.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

NullPointerException	if buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.268.3.18 void decaf::util::zip::Inflater::setInput (const std::vector< unsigned char > & *buffer,* int *offset,* int *length*)

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1125) returns true indicating that more input data is required.

Parameters

buffer	The Buffer to read in for decompression.
offset	The position in the Buffer to start reading from.
length	The number of bytes to read from the input buffer.

Exceptions

IndexOutOfBounds-Exception	if the offset + length > size of the buffer.
IllegalStateException	if in the end state.

6.268.3.19 void decaf::util::zip::Inflater::setInput (const std::vector< unsigned char > & *buffer*)

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1125) returns true indicating that more input data is required.

Parameters

buffer	The Buffer to read in for decompression.

Exceptions

IllegalStateException	if in the end state.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Inflater.h**

6.269 decaf::util::zip::InflaterInputStream Class Reference

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

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

Inheritance diagram for decaf::util::zip::InflaterInputStream:

Public Member Functions

- **InflaterInputStream** (**decaf::io::InputStream** ∗**inputStream**, bool **own**=false)

  Create an instance of this class with a default inflater and buffer size.
- **InflaterInputStream** (**decaf::io::InputStream** ∗**inputStream**, **Inflater** ∗**inflater**, bool **own**=false, bool **ownInflater**=false)

  *Creates a new **InflaterInputStream** (p. 1128) with a user supplied **Inflater** (p. 1121) and a default buffer size.*
- **InflaterInputStream** (**decaf::io::InputStream** ∗**inputStream**, **Inflater** ∗**inflater**, int bufferSize, bool **own**=false, bool **ownInflater**=false)

  *Creates a new DeflateOutputStream with a user supplied **Inflater** (p. 1121) and specified buffer size.*
- virtual ∼**InflaterInputStream** ()
- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual void **close** ()

  *Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.*
  The default implementation of this method does nothing.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs while closing the **InputStream** (p. 1134).*

- virtual long long **skip** (long long num)

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.
The skip method of ***InputStream*** *(*p. *1134) creates a byte array and then repeatedly reads into it until num 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. 1198)	*if an I/O error occurs.*
UnsupportedOperation-Exception	*if the concrete stream class does not support skipping bytes.*

- 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.
  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.
  Calling mark on a closed stream instance should have no effect.
  The default implementation of this method does nothing.

  Parameters

readLimit	*The max bytes read before marked position is invalid.*

- virtual void **reset** ()

  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. *1198) might be thrown.*
  - *If such an* ***IOException*** *(*p. *1198) 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. *1198).*
  - *If an* ***IOException*** *(*p. *1198) 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.*

  The default implementation of this method throws an ***IOException*** *(*p. *1198).*

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual bool **markSupported** () **const**

  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.
  The default implementation of this method returns false.

  Returns

  true if this stream instance supports marks

Protected Member Functions

- virtual void **fill** ()

  Fills the input buffer with the next chunk of data.

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int **length**)

Protected Attributes

- **Inflater** * **inflater**

  *The **Inflater** (p. 1121) instance to use.*
- std::vector< unsigned char > **buff**

  The buffer to hold chunks of data read from the stream before inflation.
- int **length**

  The amount of data currently stored in the input buffer.
- bool **ownInflater**
- bool **atEOF**

Static Protected Attributes

- static **const** int **DEFAULT_BUFFER_SIZE**

6.269.1 Detailed Description

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

Since

1.0

6.269.2 Constructor & Destructor Documentation

6.269.2.1 decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * *inputStream,* bool *own =* `false`)

Create an instance of this class with a default inflater and buffer size.

Parameters

inputStream	The InputStream instance to wrap.
own	Should this Filter take ownership of the InputStream pointer (defaults to false).

6.269.2.2 decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * *inputStream,* Inflater * *inflater,* bool *own =* `false`, bool *ownInflater =* `false`)

Creates a new **InflaterInputStream** (p. 1128) with a user supplied **Inflater** (p. 1121) and a default buffer size.

When the user supplied a **Inflater** (p. 1121) instance the **InflaterInputStream** (p. 1128) does not take ownership of the **Inflater** (p. 1121) pointer unless the ownInflater parameter is set to true, otherwise the caller is still responsible for deleting the **Inflater** (p. 1121).

Parameters

inputStream	The InputStream instance to wrap.
inflater	The user supplied **Inflater** (p. 1121) to use for decompression. (
own	Should this filter take ownership of the InputStream pointer (default is false).
ownInflater	Should the filter take ownership of the passed **Inflater** (p. 1121) object (default is false).

Exceptions

NullPointerException	if the **Inflater** (p. 1121) given is NULL.

6.269.2.3 decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream * *inputStream,* Inflater * *inflater,* int *bufferSize,* bool *own* = `false`, bool *ownInflater* = `false`)

Creates a new DeflateOutputStream with a user supplied **Inflater** (p. 1121) and specified buffer size.

When the user supplied a **Inflater** (p. 1121) instance the **InflaterInputStream** (p. 1128) does not take ownership of the **Inflater** (p. 1121) pointer unless the ownInflater parameter is set to true, otherwise the caller is still responsible for deleting the **Inflater** (p. 1121).

Parameters

inputStream	The InputStream instance to wrap.
inflater	The user supplied **Inflater** (p. 1121) to use for decompression.
bufferSize	The size of the input buffer.
own	Should this filter take ownership of the InputStream pointer (default is false).
ownInflater	Should the filter take ownership of the passed **Inflater** (p. 1121) object (default is false).

Exceptions

NullPointerException	if the **Inflater** (p. 1121) given is NULL.
IllegalArgumentException	if the bufferSize value is zero.

6.269.2.4 virtual decaf::util::zip::InflaterInputStream::~InflaterInputStream ( ) [virtual]

6.269.3 Member Function Documentation

6.269.3.1 virtual int decaf::util::zip::InflaterInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Until EOF this method always returns 1, thereafter it always returns 0.

Reimplemented from **decaf::io::FilterInputStream** (p. 1034).

6.269.3.2 virtual void decaf::util::zip::InflaterInputStream::close ( ) [virtual]

Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs while closing the **InputStream** (p. 1134).

Closes any resources associated with this **InflaterInputStream** (p. 1128).

Reimplemented from **decaf::io::FilterInputStream** (p. 1034).

6.269.3.3 virtual int decaf::util::zip::InflaterInputStream::doReadArrayBounded (unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.269.3.4 virtual int decaf::util::zip::InflaterInputStream::doReadByte ( ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.269.3.5 virtual void decaf::util::zip::InflaterInputStream::fill ( ) [protected, virtual]

Fills the input buffer with the next chunk of data.

Exceptions

IOException	if an I/O error occurs.

6.269.3.6 virtual void decaf::util::zip::InflaterInputStream::mark (int *readLimit*) [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.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.

Does nothing.

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.269.3.7 virtual bool decaf::util::zip::InflaterInputStream::markSupported ( ) const [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.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Always returns false.

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.269.3.8 virtual void decaf::util::zip::InflaterInputStream::reset ( ) [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. 1198) might be thrown.
- If such an **IOException** (p. 1198) 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. 1198).
- If an **IOException** (p. 1198) 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.

The default implementation of this method throws an **IOException** (p. 1198).

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Always throws an IOException when called.

Reimplemented from **decaf::io::FilterInputStream** (p. 1036).

6.269.3.9 virtual long long decaf::util::zip::InflaterInputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Skips the specified amount of uncompressed input data.

Reimplemented from **decaf::io::FilterInputStream** (p. 1036).

6.269.4 Field Documentation

6.269.4.1 bool decaf::util::zip::InflaterInputStream::atEOF [protected]

6.269.4.2 std::vector<unsigned char> decaf::util::zip::InflaterInputStream::buff [protected]

The buffer to hold chunks of data read from the stream before inflation.

6.269.4.3 const int decaf::util::zip::InflaterInputStream::DEFAULT_BUFFER_SIZE [static, protected]

6.269.4.4 Inflater* decaf::util::zip::InflaterInputStream::inflater [protected]

The **Inflater** (p. 1121) instance to use.

6.269.4.5 int decaf::util::zip::InflaterInputStream::length [protected]

The amount of data currently stored in the input buffer.

6.269.4.6 bool decaf::util::zip::InflaterInputStream::ownInflater [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**InflaterInputStream.h**

6.270 decaf::io::InputStream Class Reference

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

`#include <src/main/decaf/io/InputStream.h>`

Inheritance diagram for decaf::io::InputStream:

Public Member Functions

- **InputStream** ()
- virtual ~**InputStream** ()
- virtual void **close** ()

  *Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this 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** ()

  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 int **available** () **const**

  Indicates the number of bytes available.

- virtual int **read** ()

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char ∗buffer, int size)

  Reads up to size bytes of data from the input stream into an array of bytes.

- virtual int **read** (unsigned char ∗buffer, int size, int offset, int length)

  Reads up to length bytes of data from the input stream into an array of bytes.

- virtual long long **skip** (long long num)

  Skips over and discards n bytes of data from this input stream.

- virtual std::string **toString** () **const**

  Output a String representation of this object.

- virtual void **lock** ()

  Locks the object.

- virtual bool **tryLock** ()

  Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

  Unlocks the object.

- virtual void **wait** ()

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

- virtual void **wait** (long long millisecs)

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

- virtual void **wait** (long long millisecs, int nanos)

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

- virtual void **notify** ()

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

- virtual void **notifyAll** ()

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

Protected Member Functions

- virtual int **doReadByte** ()=0
- virtual int **doReadArray** (unsigned char ∗buffer, int size)
- virtual int **doReadArrayBounded** (unsigned char ∗buffer, int size, int offset, int length)

6.270.1 Detailed Description

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

Since

1.0

6.270.2 Constructor & Destructor Documentation

6.270.2.1 decaf::io::InputStream::InputStream ( )

6.270.2.2 virtual decaf::io::InputStream::~InputStream ( ) [virtual]

6.270.3 Member Function Documentation

6.270.3.1 virtual int decaf::io::InputStream::available ( ) const [inline, virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs.

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 530), **decaf::io::PushbackInputStream** (p. 1718), **decaf::util::zip::InflaterInputStream** (p. 1131), **decaf::io::BufferedInputStream** (p. 459), **decaf::io::BlockingByteArrayInputStream** (p. 415), **decaf::io::FilterInputStream** (p. 1034), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2083), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 1581), and **decaf::internal::io::StandardInputStream** (p. 1961).

6.270.3.2 virtual void decaf::io::InputStream::close ( ) [virtual]

Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs while closing the **InputStream** (p. 1134).

Implements **decaf::io::Closeable** (p. 633).

Reimplemented in **decaf::util::zip::InflaterInputStream** (p. 1131), **decaf::io::BufferedInputStream** (p. 459), **decaf::io::BlockingByteArrayInputStream** (p. 416), **decaf::io::FilterInputStream** (p. 1034), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2084), and **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 1581).

6.270.3.3 virtual int decaf::io::InputStream::doReadArray (unsigned char * *buffer*, int *size*) [protected, virtual]

Reimplemented in **decaf::io::FilterInputStream** (p. 1035).

6.270.3.4 virtual int decaf::io::InputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 530), **decaf::util::zip::InflaterInputStream** (p. 1132), **decaf::io::PushbackInputStream** (p. 1719), **decaf::io::BufferedInputStream** (p. 459), **decaf::io::FilterInputStream** (p. 1035), **decaf::io::BlockingByteArrayInputStream** (p. 416), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2084), **decaf::util::zip::CheckedInputStream** (p. 626), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 1582), and **activemq::io::LoggingInputStream** (p. 1323).

6.270.3.5 virtual int decaf::io::InputStream::doReadByte ( ) [protected, pure virtual]

Implemented in **decaf::io::ByteArrayInputStream** (p. 531), **decaf::util::zip::InflaterInputStream** (p. 1132), **decaf::io::PushbackInputStream** (p. 1719), **decaf::io::BufferedInputStream** (p. 460), **decaf::io::FilterInputStream** (p. 1035), **decaf::io::BlockingByteArrayInputStream** (p. 416), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2084), **decaf::util::zip::CheckedInputStream** (p. 626), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 1582), **activemq::io::LoggingInputStream** (p. 1323), and **decaf::internal::io::StandardInputStream** (p. 1962).

6.270.3.6 virtual void decaf::io::InputStream::lock ( ) [inline, virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

6.270.3.7 virtual void decaf::io::InputStream::mark (int *readLimit*) [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.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 531), **decaf::io::PushbackInputStream** (p. 1719), **decaf::util::zip::InflaterInputStream** (p. 1132), **decaf::io::BufferedInputStream** (p. 460), and **decaf::io::FilterInputStream** (p. 1035).

6.270.3.8 virtual bool decaf::io::InputStream::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.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 531), **decaf::io::PushbackInputStream** (p. 1719), **decaf::util::zip::InflaterInputStream** (p. 1132), **decaf::io::BufferedInputStream** (p. 460), and **decaf::io::FilterInputStream** (p. 1035).

6.270.3.9 virtual void decaf::io::InputStream::notify ( ) [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. 2047).

6.270.3.10 virtual void decaf::io::InputStream::notifyAll ( ) [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. 2048).

6.270.3.11 virtual int decaf::io::InputStream::read ( ) [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.

The default implementation of this method calls the internal virtual method doReadByte which is a pure virtual method and must be overridden by all subclasses.

Returns

The next byte or -1 if the end of stream is reached.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

6.270.3.12 virtual int decaf::io::InputStream::read (unsigned char * *buffer,* int *size*) [virtual]

Reads up to size bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as size bytes, but a smaller number may be read. 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 size 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 b.

This method called the protected virtual method doReadArray which by default is the same as calling read(buffer, size, 0, size). Subclasses can customize the behavior of this method by overriding the doReadArray method to provide a better performing read operation.

Parameters

buffer	The target buffer to write the read in data to.
size	The size in bytes of the target buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.
NullPointerException	if buffer passed is NULL.

6.270.3.13 virtual int decaf::io::InputStream::read (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [virtual]

Reads up to length bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as length bytes, but a smaller number may be read. 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 length 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 b.

The first byte read is stored into element b[off], the next one into b[off+1], and so on. The number of bytes read is, at most, equal to length. Let k be the number of bytes actually read; these bytes will be stored in elements b[off] through b[off+k-1], leaving elements b[offset+k] through b[offset+length-1] unaffected.

In every case, elements b[0] through b[offset] and elements b[offset+length] through b[b.length-1] are unaffected.

This method called the protected virtual method doReadArrayBounded which simply calls the method **doReadByte()** (p. 1137) repeatedly. If the first such call results in an **IOException** (p. 1198), that exception is returned. If any subsequent call to **doReadByte()** (p. 1137) results in a **IOException** (p. 1198), the exception is caught and treated as if it were end of file; the bytes read up to that point are stored into the buffer and the number of bytes read before the exception occurred is returned. The default implementation of this method blocks until the requested amount of input data has been read, end of file is detected, or an exception is thrown. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

buffer	The target buffer to write the read in data to.
size	The size in bytes of the target buffer.
offset	The position in the buffer to start inserting the read in data.
length	The maximum number of bytes that should be read into buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs.
NullPointerException	if buffer passed is NULL.
IndexOutOfBounds-Exception	if length $>$ size - offset.

6.270.3.14 virtual void decaf::io::InputStream::reset ( ) [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. 1198) might be thrown.
- If such an **IOException** (p. 1198) 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. 1198).
- If an **IOException** (p. 1198) 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.

The default implementation of this method throws an **IOException** (p. 1198).

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs.

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 531), **decaf::io::PushbackInputStream** (p. 1719), **decaf::util::zip::InflaterInputStream** (p. 1133), **decaf::io::BufferedInputStream** (p. 460), and **decaf::io::FilterInputStream** (p. 1036).

6.270.3.15 virtual long long decaf::io::InputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Reimplemented in **decaf::io::ByteArrayInputStream** (p. 532), **decaf::io::PushbackInputStream** (p. 1720), **decaf::util::zip::InflaterInputStream** (p. 1133), **decaf::io::BufferedInputStream** (p. 461), **decaf::io::BlockingByteArrayInputStream** (p. 416), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 2084), **decaf::io::FilterInputStream** (p. 1036), **decaf::util::zip::CheckedInputStream** (p. 626), and **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 1582).

6.270.3.16 virtual std::string decaf::io::InputStream::toString ( ) const [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

6.270.3.17 virtual bool decaf::io::InputStream::tryLock ( ) [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. 2048).

6.270.3.18 virtual void decaf::io::InputStream::unlock ( ) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

6.270.3.19 virtual void decaf::io::InputStream::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.270.3.20 virtual void decaf::io::InputStream::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.270.3.21 virtual void decaf::io::InputStream::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InputStream.h**

6.271 decaf::io::InputStreamReader Class Reference

An **InputStreamReader** (p. 1142) is a bridge from byte streams to character streams.

`#include <src/main/decaf/io/InputStreamReader.h>`

Inheritance diagram for decaf::io::InputStreamReader:

Public Member Functions

- **InputStreamReader** (**InputStream** ∗stream, bool own=false)

  *Create a new **InputStreamReader** (p. 1142) that wraps the given **InputStream** (p. 1134).*
- virtual ∼**InputStreamReader** ()
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- virtual bool **ready** () **const**

  Tells whether this stream is ready to be read.

Protected Member Functions

- virtual int **doReadArrayBounded** (char ∗buffer, int size, int offset, int length)

  Override this method to customize the functionality of the method read(unsigned char∗ buffer, int size, int offset, int length).
- virtual void **checkClosed** () **const**

6.271.1 Detailed Description

An **InputStreamReader** (p. 1142) is a bridge from byte streams to character streams.

For top efficiency, consider wrapping an **InputStreamReader** (p. 1142) within a BufferedReader. For example:

BufferedReader∗ in = new BufferedReader(new **InputStreamReader** (p. 1142)(System.in, false), true);

See also

OutputStreamWriter (p. 1606)

Since

1.0

6.271.2 Constructor & Destructor Documentation

6.271.2.1 decaf::io::InputStreamReader::InputStreamReader (InputStream * *stream,* bool *own* = false)

Create a new **InputStreamReader** (p. 1142) that wraps the given **InputStream** (p. 1134).

Parameters

stream	The **InputStream** (p. 1134) to read from. (cannot be null).
own	Does this object own the passed **InputStream** (p. 1134) (defaults to false).

Exceptions

NullPointerException	if the passed **InputStream** (p. 1134) is NULL.

6.271.2.2 virtual decaf::io::InputStreamReader::~InputStreamReader ( ) [virtual]

6.271.3 Member Function Documentation

6.271.3.1 virtual void decaf::io::InputStreamReader::checkClosed ( ) const [protected, virtual]

6.271.3.2 virtual void decaf::io::InputStreamReader::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(p. 1198)*	if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 633).

6.271.3.3 virtual int decaf::io::InputStreamReader::doReadArrayBounded (char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Override this method to customize the functionality of the method read(unsigned char* buffer, int size, int offset, int length).

All subclasses must override this method to provide the basic **Reader** (p. 1734) functionality.

Implements **decaf::io::Reader** (p. 1735).

6.271.3.4 virtual bool decaf::io::InputStreamReader::ready ( ) const [virtual]

Tells whether this stream is ready to be read.

Returns

True if the next **read()** (p. 1737) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::Reader** (p. 1738).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InputStreamReader.h**

6.272 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** (int size, bool readOnly=false)

  *Creates a **IntArrayBuffer** (p. 1144) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **IntArrayBuffer** (int ∗**array**, int size, int offset, int length, bool readOnly=false)

  *Creates a **IntArrayBuffer** (p. 1144) object that wraps the given array.*
- **IntArrayBuffer** (**const decaf::lang::Pointer**< **ByteArrayAdapter** > &**array**, int offset, int length, bool readOnly=false)

  Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **IntArrayBuffer** (**const IntArrayBuffer** &other)

  *Create a **IntArrayBuffer** (p. 1144) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ∼**IntArrayBuffer** ()
- virtual int ∗ **array** ()

  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. 451).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual int **arrayOffset** ()

  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. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual **IntBuffer** ∗ **asReadOnlyBuffer** () **const**

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.

- virtual **IntBuffer** & **compact** ()

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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1152)*

Exceptions

ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **IntBuffer** ∗ **duplicate** ()

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. 451) which the caller owns.*

- virtual int **get** ()

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. 474)	*if there no more data to return.*

- virtual int **get** (int index) **const**

Absolute get method.
Reads the value at the given index.

Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual bool **hasArray** () **const**

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. Subclasses 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.

- virtual bool **isReadOnly** () **const**

  Tells whether or not this buffer is read-only.

  Returns

  true if, and only if, this buffer is read-only.

- virtual **IntBuffer** & **put** (int value)

  Writes the given integer into this buffer at the current position, and then increments the position.

  Parameters

value	*The integer value to be written.*

  Returns

  a reference to this buffer.

  Exceptions

BufferOverflowException (p. 472)	*if this buffer's current position is not smaller than its limit.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **IntBuffer** & **put** (int index, int value)

  Writes the given ints into this buffer at the given index.

  Parameters

index	*The position in the **Buffer** (p. 451) 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, or the index is negative.*
ReadOnlyBufferException (p. 1739)	*- If this buffer is read-only.*

- virtual **IntBuffer** ∗ **slice** () **const**

  *Creates a new **IntBuffer** (p. 1152) 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. 1152) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **IntArrayBuffer** (p. 1144) as Read-Only.*

6.272.1 Constructor & Destructor Documentation

6.272.1.1 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int *size*, bool *readOnly* = `false`)

Creates a **IntArrayBuffer** (p. 1144) 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

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArguementException	if the capacity value is negative.

6.272.1.2 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int * *array*, int *size*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a **IntArrayBuffer** (p. 1144) 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	The actual array to wrap.
size	The size of the given array.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if offset is greater than array capacity.

6.272.1.3 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **IntArrayBuffer** (p. 1144) will be that of the remaining capacity of the passed buffer.

Parameters

array	The ByteArrayAdapter to wrap.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if array is NULL
IndexOutOfBounds-Exception	if offset + length is greater than array size.

6.272.1.4 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const IntArrayBuffer & *other*)

Create a **IntArrayBuffer** (p. 1144) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.

Parameters

other	The **IntArrayBuffer** (p. 1144) this one is to mirror.

6.272.1.5 virtual decaf::internal::nio::IntArrayBuffer::~IntArrayBuffer ( ) [virtual]

6.272.2 Member Function Documentation

6.272.2.1 virtual int* decaf::internal::nio::IntArrayBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1155).

6.272.2.2 virtual int decaf::internal::nio::IntArrayBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1155).

6.272.2.3 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. 1155).

6.272.2.4 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1152)

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1156).

6.272.2.5 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** (p. 451) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1156).

6.272.2.6 virtual int decaf::internal::nio::IntArrayBuffer::get ( ) [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. 474)*	if there no more data to return.

Implements **decaf::nio::IntBuffer** (p. 1156).

6.272.2.7 virtual int decaf::internal::nio::IntArrayBuffer::get (int *index*) const [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the int is to be read.

Returns

the int that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::IntBuffer** (p. 1157).

6.272.2.8 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. Subclasses 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. 1158).

6.272.2.9 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.

Implements **decaf::nio::Buffer** (p. 454).

6.272.2.10 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int *value*) [virtual]

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

value	The integer value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1159).

6.272.2.11 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int *index,* int *value*) [virtual]

Writes the given ints into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The ints to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBufferException *(p. 1739)*	- If this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 1160).

6.272.2.12 virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **IntArrayBuffer** (p. 1144) as Read-Only.

Parameters

value	Boolean value, true if this buffer is to be read-only, false otherwise.

6.272.2.13 virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice ( ) const [virtual]

Creates a new **IntBuffer** (p. 1152) 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. 1152) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1160).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**IntArrayBuffer.h**

6.273 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

  Returns the int array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0

  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

  Compacts this buffer.
- virtual **IntBuffer** * **duplicate** ()=0

  Creates a new int buffer that shares this buffer's content.
- virtual int **get** ()=0

  Relative get method.
- virtual int **get** (int index) **const** =0

  Absolute get method.
- **IntBuffer** & **get** (std::vector< int > buffer)

  Relative bulk get method.
- **IntBuffer** & **get** (int *buffer, int size, int offset, int length)

  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)

  This method transfers the ints remaining in the given source buffer into this buffer.
- **IntBuffer** & **put** (**const** int *buffer, int size, int offset, int length)

  This method transfers ints into this buffer from the given source array.
- **IntBuffer** & **put** (std::vector< int > &buffer)

  This method transfers the entire content of the given source ints array into this buffer.
- virtual **IntBuffer** & **put** (int value)=0

Writes the given integer into this buffer at the current position, and then increments the position.

- virtual **IntBuffer** & **put** (int index, int value)=0

  Writes the given ints into this buffer at the given index.

- virtual **IntBuffer** * **slice** () **const** =0

  *Creates a new **IntBuffer** (p. 1152) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (**const IntBuffer** &value) **const**

- virtual bool **equals** (**const IntBuffer** &value) **const**

- virtual bool **operator==** (**const IntBuffer** &value) **const**

- virtual bool **operator<** (**const IntBuffer** &value) **const**

Static Public Member Functions

- static **IntBuffer** * **allocate** (int **capacity**)

  Allocates a new Double buffer.

- static **IntBuffer** * **wrap** (int ***array**, int size, int offset, int length)

  *Wraps the passed buffer with a new **IntBuffer** (p. 1152).*

- static **IntBuffer** * **wrap** (std::vector< int > &buffer)

  *Wraps the passed STL int Vector in a **IntBuffer** (p. 1152).*

Protected Member Functions

- **IntBuffer** (int **capacity**)

  *Creates a **IntBuffer** (p. 1152) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.273.1 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.

6.273.2 Constructor & Destructor Documentation

6.273.2.1 decaf::nio::IntBuffer::IntBuffer (int *capacity*) [protected]

Creates a **IntBuffer** (p. 1152) 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	The size and limit of the **Buffer** (p. 451) in integers.

Exceptions

IllegalArguementException	if capacity is negative.

6.273.2.2 virtual decaf::nio::IntBuffer::~IntBuffer ( ) [inline, virtual]

6.273.3 Member Function Documentation

6.273.3.1 static IntBuffer* decaf::nio::IntBuffer::allocate (int *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 integers.

Returns

the **IntBuffer** (p. 1152) that was allocated, caller owns.

6.273.3.2 virtual int* decaf::nio::IntBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1148).

6.273.3.3 virtual int decaf::nio::IntBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1149).

6.273.3.4 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. 1149).

6.273.3.5 virtual IntBuffer& decaf::nio::IntBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1152)

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1149).

6.273.3.6 virtual int decaf::nio::IntBuffer::compareTo (const IntBuffer & *value*) const [virtual]

6.273.3.7 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. 451) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1150).

6.273.3.8 virtual bool decaf::nio::IntBuffer::equals (const IntBuffer & *value*) const [virtual]

6.273.3.9 virtual int decaf::nio::IntBuffer::get ( ) [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. 474)*	if there no more data to return.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1150).

6.273.3.10 virtual int decaf::nio::IntBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the int is to be read.

Returns

the int that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1150).

6.273.3.11 IntBuffer& decaf::nio::IntBuffer::get (std::vector< int > *buffer*)

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. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length ints remaining in this buffer.

6.273.3.12 IntBuffer& decaf::nio::IntBuffer::get (int * *buffer*, int *size*, int *offset*, int *length*)

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. 456), then no bytes are transferred and a **BufferUnderflowException** (p. 474) 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	The pointer to an allocated buffer to fill.
size	The size of the buffer that was passed in.
offset	The position in the buffer to start filling.
length	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length ints remaining in this buffer.
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.273.3.13 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. Subclasses 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. 1151).

6.273.3.14 virtual bool decaf::nio::IntBuffer::operator< (const IntBuffer & *value*) const [virtual]

6.273.3.15 virtual bool decaf::nio::IntBuffer::operator== (const IntBuffer & *value*) const [virtual]

6.273.3.16 IntBuffer& decaf::nio::IntBuffer::put (IntBuffer & *src*)

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. 456), then no ints are transferred and a **BufferOverflowException** (p. 472) 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. 472)*	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. 1739)*	if this buffer is read-only.

6.273.3.17 IntBuffer& decaf::nio::IntBuffer::put (const int * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no ints are transferred and a **BufferOverflowException** (p. 472) 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 integers are to be read.
size	The size of the buffer passed.
offset	The offset within the array of the first char to be read.
length	The number of integers to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.273.3.18 IntBuffer& decaf::nio::IntBuffer::put (std::vector< int > & *buffer*)

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. 1152).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

6.273.3.19 virtual IntBuffer& decaf::nio::IntBuffer::put (int *value*) [pure virtual]

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

value	The integer value to be written.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1151).

6.273.3.20 virtual IntBuffer& decaf::nio::IntBuffer::put (int *index,* int *value*) [pure virtual]

Writes the given ints into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The ints to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
ReadOnlyBufferException *(p. 1739)*	- If this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1151).

6.273.3.21 virtual IntBuffer* decaf::nio::IntBuffer::slice ( ) const [pure virtual]

Creates a new **IntBuffer** (p. 1152) 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. 1152) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1152).

6.273.3.22 virtual std::string decaf::nio::IntBuffer::toString ( ) const [virtual]

Returns

a std::string describing this object

6.273.3.23 static IntBuffer* decaf::nio::IntBuffer::wrap (int * *array,* int *size,* int *offset,* int *length*) [static]

Wraps the passed buffer with a new **IntBuffer** (p. 1152).

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.
size	The size of the passed in array.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new **IntBuffer** (p. 1152) that is backed by buffer, caller owns.

Exceptions

NullPointerException	if the array pointer is NULL.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.273.3.24 static IntBuffer* decaf::nio::IntBuffer::wrap (std::vector< int > & *buffer*) [static]

Wraps the passed STL int Vector in a **IntBuffer** (p. 1152).

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. 1152) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**IntBuffer.h**

6.274 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)

  *Constructs a new **Integer** (p. 1161) and attempts to convert the given string to an int value, assigning it to the new object is successful or throwing a NumberFormatException if the string is not a properly formatted int.*
- virtual ~**Integer** ()
- virtual int **compareTo** (**const Integer** &i) **const**

  *Compares this **Integer** (p. 1161) 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. 1161) 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)

  *Decodes a **String** (p. 2031) into a **Integer** (p. 1161).*
- 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)

  Parses the string argument as a signed int in the radix specified by the second argument.
- static int **parseInt** (**const** std::string &s)

  Parses the string argument as a signed decimal int.
- static **Integer valueOf** (int value)

  *Returns a **Integer** (p. 1161) instance representing the specified int value.*
- static **Integer valueOf** (**const** std::string &value)

  *Returns a **Integer** (p. 1161) object holding the value given by the specified std::string.*
- static **Integer valueOf** (**const** std::string &value, int radix)

  *Returns a **Integer** (p. 1161) 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** (p. 2031) 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.

6.274.1 Constructor & Destructor Documentation

6.274.1.1 decaf::lang::Integer::Integer (int *value*)

Parameters

value	The primitive value to wrap in an **Integer** (p. 1161) instance.

6.274.1.2 decaf::lang::Integer::Integer (const std::string & *value*)

Constructs a new **Integer** (p. 1161) and attempts to convert the given string to an int value, assigning it to the new object is successful or throwing a NumberFormatException if the string is not a properly formatted int.

Parameters

value	The string to convert to a primitive type to wrap.

Exceptions

NumberFormatException	if the string is not a a valid integer.

6.274.1.3 virtual decaf::lang::Integer::~Integer ( ) [inline, virtual]

6.274.2 Member Function Documentation

6.274.2.1 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.

6.274.2.2 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.

Reimplemented from **decaf::lang::Number** (p. 1544).

6.274.2.3 virtual int decaf::lang::Integer::compareTo (const Integer & *i*) const [virtual]

Compares this **Integer** (p. 1161) instance with another.

Parameters

i	- the **Integer** (p. 1161) 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**< **Integer** > (p. 687).

6.274.2.4 virtual int decaf::lang::Integer::compareTo (const int & *i*) const [virtual]

Compares this **Integer** (p. 1161) instance with another.

Parameters

i	- the **Integer** (p. 1161) 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. 687).

6.274.2.5 static Integer decaf::lang::Integer::decode (const std::string & *value*) [static]

Decodes a **String** (p. 2031) into a **Integer** (p. 1161).

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.parseInteger 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** (p. 2031) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value	- The string to decode

Returns

a **Integer** (p. 1161) object containing the decoded value

Exceptions

NumberFomatException	if the string is not formatted correctly.

6.274.2.6 virtual double decaf::lang::Integer::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.274.2.7 bool decaf::lang::Integer::equals (const Integer & *i*) const [inline, virtual]

Parameters

i	- the **Integer** (p. 1161) object to compare against.

Returns

true if the two **Integer** (p. 1161) Objects have the same value.

Implements **decaf::lang::Comparable**< **Integer** > (p. 688).

6.274.2.8 bool decaf::lang::Integer::equals (const int & *i*) const [inline, virtual]

Parameters

i	- the **Integer** (p. 1161) object to compare against.

Returns

true if the two **Integer** (p. 1161) Objects have the same value.

Implements **decaf::lang::Comparable**< **int** > (p. 688).

6.274.2.9 virtual float decaf::lang::Integer::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.274.2.10 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.

6.274.2.11 virtual int decaf::lang::Integer::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.274.2.12 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.

Implements **decaf::lang::Number** (p. 1545).

6.274.2.13 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.

6.274.2.14 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:

```
* floor( log2(x)) = 31 - numberOfLeadingZeros(x)
* ceil( log2(x)) = 32 - 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.

6.274.2.15 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.

6.274.2.16 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.

Implements **decaf::lang::Comparable< Integer >** (p. 688).

6.274.2.17 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. 688).

6.274.2.18 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.

Implements **decaf::lang::Comparable< Integer >** (p. 688).

6.274.2.19 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. 688).

6.274.2.20 static int decaf::lang::Integer::parseInt (const std::string & *s*, int *radix*) [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. 595) 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. 599) or larger than **Character.MAX_RADIX** (p. 599).
- 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** (p. 2031) 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** (p. 2031) does not contain a parsable int.

6.274.2.21 static int decaf::lang::Integer::parseInt (const std::string & *s*) [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 parseInteger(const std::string, int) method.

Parameters

s	- **String** (p. 2031) to convert to a int

Returns

the converted int value

Exceptions

NumberFormatException	if the string is not a int.

6.274.2.22 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.

6.274.2.23 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.

6.274.2.24 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.

6.274.2.25 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.

6.274.2.26 virtual short decaf::lang::Integer::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1545).

6.274.2.27 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.

6.274.2.28 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

6.274.2.29 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

6.274.2.30 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

6.274.2.31 std::string decaf::lang::Integer::toString ( ) const

Returns

this **Integer** (p. 1161) Object as a **String** (p. 2031) Representation

Referenced by decaf::util::ArrayList< E >::toString().

6.274.2.32 static std::string decaf::lang::Integer::toString (int *value*) [static]

Converts the int to a **String** (p. 2031) representation.

Parameters

value	The int to convert to a `std::string` instance.

Returns

string representation

6.274.2.33 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. 599) or larger than **Character.MAX_RADIX** (p. 599), 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.

6.274.2.34 static Integer decaf::lang::Integer::valueOf (int *value*) [inline, static]

Returns a **Integer** (p. 1161) instance representing the specified int value.

Parameters

value	- the int to wrap

Returns

the new **Integer** (p. 1161) object wrapping value.

6.274.2.35 static Integer decaf::lang::Integer::valueOf (const std::string & *value*) [static]

Returns a **Integer** (p. 1161) 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. 1161) object that represents the int value specified by the string.

Parameters

value	- std::string to parse as base 10

Returns

new **Integer** (p. 1161) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a decimal int.

6.274.2.36 static Integer decaf::lang::Integer::valueOf (const std::string & *value,* int *radix*) [static]

Returns a **Integer** (p. 1161) 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. 1161) 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. 1161) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a valid int.

6.274.3 Field Documentation

6.274.3.1 const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF [static]

The maximum value that the primitive type can hold.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo().

6.274.3.2 const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000 [static]

The minimum value that the primitive type can hold.

6.274.3.3 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**

6.275 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **IntegerResponse** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 Attributes

- int **result**

6.275.1 Constructor & Destructor Documentation

6.275.1.1 activemq::commands::IntegerResponse::IntegerResponse ( )

6.275.1.2 virtual activemq::commands::IntegerResponse::~IntegerResponse ( ) [virtual]

6.275.2 Member Function Documentation

6.275.2.1 virtual IntegerResponse* activemq::commands::IntegerResponse::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1782).

6.275.2.2 virtual void activemq::commands::IntegerResponse::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::Response** (p. 1782).

6.275.2.3 virtual bool activemq::commands::IntegerResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.275.2.4 virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::Response** (p. 1783).

6.275.2.5 virtual int activemq::commands::IntegerResponse::getResult ( ) const [virtual]

6.275.2.6 virtual void activemq::commands::IntegerResponse::setResult (int *result*) [virtual]

6.275.2.7 virtual std::string activemq::commands::IntegerResponse::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 1783).

6.275.3 Field Documentation

6.275.3.1 const unsigned char activemq::commands::IntegerResponse::ID_INTEGERRESPONSE = 34 [static]

6.275.3.2 int activemq::commands::IntegerResponse::result [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**IntegerResponse.h**

6.276 activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1177).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/IntegerResponseMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual ~**IntegerResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.276.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1177).

NOTE!: This file is 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::generated::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]

6.276.2.2 virtual activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::~IntegerResponseMarshaller ( ) [inline, virtual]

6.276.3 Member Function Documentation

6.276.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1789).

6.276.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.276.3.3 virtual void activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.276.3.4 virtual void activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1790).

6.276.3.5 virtual int activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.276.3.6 virtual void activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1791).

6.276.3.7 virtual void activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::ResponseMarshaller** (p. 1792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**IntegerResponseMarshaller.h**

6.277 internal_state Struct Reference

`#include <src/main/decaf/internal/util/zip/deflate.h>`

Data Fields

- **z_streamp strm**
- int **status**
- **Bytef** * **pending_buf**
- **ulg pending_buf_size**
- **Bytef** * **pending_out**
- **uInt pending**
- int **wrap**
- **gz_headerp gzhead**
- **uInt gzindex**
- **Byte method**
- int **last_flush**
- **uInt w_size**
- **uInt w_bits**
- **uInt w_mask**
- **Bytef** * **window**
- **ulg window_size**
- **Posf** * **prev**
- **Posf** * **head**
- **uInt ins_h**
- **uInt hash_size**
- **uInt hash_bits**
- **uInt hash_mask**
- **uInt hash_shift**
- long **block_start**
- **uInt match_length**
- **IPos prev_match**
- int **match_available**
- **uInt strstart**
- **uInt match_start**
- **uInt lookahead**
- **uInt prev_length**
- **uInt max_chain_length**
- **uInt max_lazy_match**
- int **level**
- int **strategy**
- **uInt good_match**
- int **nice_match**
- struct **ct_data_s dyn_ltree** [**HEAP_SIZE**]
- struct **ct_data_s dyn_dtree** [2 *__D_CODES__+1]
- struct **ct_data_s bl_tree** [2 *__BL_CODES__+1]
- struct **tree_desc_s l_desc**
- struct **tree_desc_s d_desc**
- struct **tree_desc_s bl_desc**

- **ush bl_count** [**MAX_BITS**+1]
- int **heap** [2 ∗**L_CODES**+1]
- int **heap_len**
- int **heap_max**
- **uch depth** [2 ∗**L_CODES**+1]
- **uchf** ∗ **l_buf**
- **ulnt lit_bufsize**
- **ulnt last_lit**
- **ushf** ∗ **d_buf**
- **ulg opt_len**
- **ulg static_len**
- **ulnt matches**
- int **last_eob_len**
- **ush bi_buf**
- int **bi_valid**
- **ulg high_water**
- int **dummy**

6.277.1 Field Documentation

6.277.1.1 ush internal_state::bi_buf

6.277.1.2 int internal_state::bi_valid

6.277.1.3 ush internal_state::bl_count[MAX_BITS+1]

6.277.1.4 struct tree_desc_s internal_state::bl_desc

6.277.1.5 struct ct_data_s internal_state::bl_tree[2 ∗BL_CODES+1]

6.277.1.6 long internal_state::block_start

6.277.1.7 ushf∗ internal_state::d_buf

6.277.1.8 struct tree_desc_s internal_state::d_desc

6.277.1.9 uch internal_state::depth[2 ∗L_CODES+1]

6.277.1.10 int internal_state::dummy

6.277.1.11 struct ct_data_s internal_state::dyn_dtree[2 ∗D_CODES+1]

6.277.1.12 struct ct_data_s internal_state::dyn_ltree[HEAP_SIZE]

6.277.1.13 ulnt internal_state::good_match

6.277.1.14 gz_headerp internal_state::gzhead

6.277.1.15 ulnt internal_state::gzindex

6.277.1.16 ulnt internal_state::hash_bits

6.277.1.17 ulnt internal_state::hash_mask

6.277.1.18 ulnt internal_state::hash_shift

6.277.1.19 ulnt internal_state::hash_size

6.277.1.20 Posf* internal_state::head

6.277.1.21 int internal_state::heap[2 *L_CODES+1]

6.277.1.22 int internal_state::heap_len

6.277.1.23 int internal_state::heap_max

6.277.1.24 ulg internal_state::high_water

6.277.1.25 ulnt internal_state::ins_h

6.277.1.26 uchf* internal_state::l_buf

6.277.1.27 struct tree_desc_s internal_state::l_desc

6.277.1.28 int internal_state::last_eob_len

6.277.1.29 int internal_state::last_flush

6.277.1.30 ulnt internal_state::last_lit

6.277.1.31 int internal_state::level

6.277.1.32 ulnt internal_state::lit_bufsize

6.277.1.33 ulnt internal_state::lookahead

6.277.1.34 int internal_state::match_available

6.277.1.35 ulnt internal_state::match_length

6.277.1.36 ulnt internal_state::match_start

6.277.1.37 ulnt internal_state::matches

6.277.1.38 ulnt internal_state::max_chain_length

6.277.1.39 ulnt internal_state::max_lazy_match

6.277.1.40 Byte internal_state::method

6.277.1.41 int internal_state::nice_match

6.277.1.42 ulg internal_state::opt_len

6.277.1.43 ulnt internal_state::pending

6.277.1.44 Bytef* internal_state::pending_buf

6.277.1.45 ulg internal_state::pending_buf_size

6.277.1.46 Bytef* internal_state::pending_out

6.277.1.47 Posf* internal_state::prev

6.277.1.48 uInt internal_state::prev_length

6.277.1.49 lPos internal_state::prev_match

6.277.1.50 ulg internal_state::static_len

6.277.1.51 int internal_state::status

6.277.1.52 int internal_state::strategy

6.277.1.53 z_streamp internal_state::strm

6.277.1.54 uInt internal_state::strstart

6.277.1.55 uInt internal_state::w_bits

6.277.1.56 uInt internal_state::w_mask

6.277.1.57 uInt internal_state::w_size

6.277.1.58 Bytef* internal_state::window

6.277.1.59 ulg internal_state::window_size

6.277.1.60 int internal_state::wrap

The documentation for this struct was generated from the following files:

- src/main/decaf/internal/util/zip/**deflate.h**
- src/main/decaf/internal/util/zip/**zlib.h**

6.278 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 1509).

`#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** ()

  Default implementation of the run method - does nothing.

6.278.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 1509).

This class processes all outbound commands and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 1784) and getting a set of Commands to send back into the **MockTransport** (p. 1509) as incoming Commands and Responses.

6.278.2 Constructor & Destructor Documentation

6.278.2.1 activemq::transport::mock::InternalCommandListener::InternalCommandListener ( )

6.278.2.2 virtual activemq::transport::mock::InternalCommandListener::~InternalCommandListener ( ) [virtual]

6.278.3 Member Function Documentation

6.278.3.1 virtual void activemq::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. 2161) deletes the command upon receipt.

Parameters

command	The received command object.

Implements **activemq::transport::TransportListener** (p. 2177).

6.278.3.2 void activemq::transport::mock::InternalCommandListener::run ( ) [virtual]

Default implementation of the run method - does nothing.

Reimplemented from **decaf::lang::Thread** (p. 2100).

6.278.3.3 void activemq::transport::mock::InternalCommandListener::setResponseBuilder (const Pointer< ResponseBuilder > & *responseBuilder*) [inline]

6.278.3.4 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**

6.279 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. 990).*
- **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 ()

6.279.1 Constructor & Destructor Documentation

6.279.1.1 decaf::lang::exceptions::InterruptedException::InterruptedException ( ) throw () [inline]

Default Constructor.

6.279.1.2 decaf::lang::exceptions::InterruptedException::InterruptedException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.279.1.3 decaf::lang::exceptions::InterruptedException::InterruptedException (const InterruptedException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.279.1.4 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

6.279.1.5 decaf::lang::exceptions::InterruptedException::InterruptedException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.279.1.6 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

6.279.1.7 virtual decaf::lang::exceptions::InterruptedException::~InterruptedException ( ) throw () [inline, virtual]

6.279.2 Member Function Documentation

6.279.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InterruptedException.h**

6.280 decaf::io::InterruptedIOException Class Reference

`#include <src/main/decaf/io/InterruptedIOException.h>`

Inheritance diagram for decaf::io::InterruptedIOException:

Public Member Functions

- **InterruptedIOException** () throw ()

  Default Constructor.
- **InterruptedIOException** (**const lang::Exception** &ex) throw ()

  Copy Constructor.
- **InterruptedIOException** (**const InterruptedIOException** &ex) throw ()

  Copy Constructor.
- **InterruptedIOException** (**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.
- **InterruptedIOException** (**const** std::exception ∗**cause**) throw ()

  Constructor.
- **InterruptedIOException** (**const** char ∗file, **const** int lineNumber, **const** char ∗msg,...) throw ()

  Constructor.
- virtual **InterruptedIOException** ∗ **clone** () **const**

  Clones this exception.
- virtual ∼**InterruptedIOException** () throw ()

6.280.1 Constructor & Destructor Documentation

6.280.1.1 decaf::io::InterruptedIOException::InterruptedIOException ( ) throw () [inline]

Default Constructor.

6.280.1.2 decaf::io::InterruptedIOException::InterruptedIOException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.280.1.3 decaf::io::InterruptedIOException::InterruptedIOException (const InterruptedIOException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.280.1.4 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

6.280.1.5 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.

6.280.1.6 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

6.280.1.7 virtual decaf::io::InterruptedIOException::~InterruptedIOException ( ) throw () [inline, virtual]

6.280.2 Member Function Documentation

6.280.2.1 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. 1200).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 1934).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

6.281 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** ()
- **InvalidClientIdException** (**const InvalidClientIdException** &ex)
- **InvalidClientIdException** (**const** std::string &message)
- **InvalidClientIdException** (**const** std::string &message, **const** std::exception ∗cause)
- **InvalidClientIdException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ∼**InvalidClientIdException** () throw ()

6.281.1 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

6.281.2 Constructor & Destructor Documentation

6.281.2.1 cms::InvalidClientIdException::InvalidClientIdException ( )

6.281.2.2 cms::InvalidClientIdException::InvalidClientIdException (const InvalidClientIdException & *ex*)

6.281.2.3 cms::InvalidClientIdException::InvalidClientIdException (const std::string & *message*)

6.281.2.4 cms::InvalidClientIdException::InvalidClientIdException (const std::string & *message,* const std::exception ∗ *cause*)

6.281.2.5 cms::InvalidClientIdException::InvalidClientIdException (const std::string & *message,* const std::exception ∗ *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.281.2.6 virtual cms::InvalidClientIdException::∼InvalidClientIdException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidClientIdException.h**

6.282 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** ()
- **InvalidDestinationException** (**const InvalidDestinationException** &ex)
- **InvalidDestinationException** (**const** std::string &message)
- **InvalidDestinationException** (**const** std::string &message, **const** std::exception ∗cause)
- **InvalidDestinationException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ∼**InvalidDestinationException** () throw ()

6.282.1 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

6.282.2 Constructor & Destructor Documentation

6.282.2.1 cms::InvalidDestinationException::InvalidDestinationException ( )

6.282.2.2 cms::InvalidDestinationException::InvalidDestinationException (const InvalidDestinationException & *ex*)

6.282.2.3 cms::InvalidDestinationException::InvalidDestinationException (const std::string & *message*)

6.282.2.4 cms::InvalidDestinationException::InvalidDestinationException (const std::string & *message,* const std::exception ∗ *cause*)

6.282.2.5 cms::InvalidDestinationException::InvalidDestinationException (const std::string & *message,* const std::exception ∗ *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.282.2.6 virtual cms::InvalidDestinationException::∼InvalidDestinationException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidDestinationException.h**

6.283 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 occured.
- virtual **InvalidKeyException** ∗ **clone** () **const**

  Clones this exception.
- virtual ∼**InvalidKeyException** () throw ()

6.283.1 Constructor & Destructor Documentation

6.283.1.1 decaf::security::InvalidKeyException::InvalidKeyException ( ) throw () [inline]

Default Constructor.

6.283.1.2 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

6.283.1.3 decaf::security::InvalidKeyException::InvalidKeyException (const InvalidKeyException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.283.1.4 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

6.283.1.5 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.

6.283.1.6 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 occured.

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

6.283.1.7 virtual decaf::security::InvalidKeyException::~InvalidKeyException ( ) throw () [inline, virtual]

6.283.2 Member Function Documentation

6.283.2.1 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. 1242).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**InvalidKeyException.h**

6.284 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 ()

6.284.1 Constructor & Destructor Documentation

6.284.1.1 decaf::nio::InvalidMarkException::InvalidMarkException ( ) throw () [inline]

Default Constructor.

6.284.1.2 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.

6.284.1.3 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.

6.284.1.4 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

6.284.1.5 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.

6.284.1.6 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

6.284.1.7 virtual decaf::nio::InvalidMarkException::~InvalidMarkException ( ) throw () [inline, virtual]

6.284.2 Member Function Documentation

6.284.2.1 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. 1101).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**InvalidMarkException.h**

6.285 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** ()
- **InvalidSelectorException** (**const InvalidSelectorException** &ex)
- **InvalidSelectorException** (**const** std::string &message)
- **InvalidSelectorException** (**const** std::string &message, **const** std::exception ∗cause)
- **InvalidSelectorException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ∼**InvalidSelectorException** () throw ()

6.285.1 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

6.285.2 Constructor & Destructor Documentation

6.285.2.1 cms::InvalidSelectorException::InvalidSelectorException ( )

6.285.2.2 cms::InvalidSelectorException::InvalidSelectorException (const InvalidSelectorException & *ex*)

6.285.2.3 cms::InvalidSelectorException::InvalidSelectorException (const std::string & *message*)

6.285.2.4 cms::InvalidSelectorException::InvalidSelectorException (const std::string & *message,* const std::exception ∗ *cause*)

6.285.2.5 cms::InvalidSelectorException::InvalidSelectorException (const std::string & *message,* const std::exception ∗ *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.285.2.6 virtual cms::InvalidSelectorException::∼InvalidSelectorException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidSelectorException.h**

6.286 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. 990).*
- **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 ()

6.286.1 Constructor & Destructor Documentation

6.286.1.1 decaf::lang::exceptions::InvalidStateException::InvalidStateException ( ) throw () [inline]

Default Constructor.

6.286.1.2 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.286.1.3 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const InvalidStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.286.1.4 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

6.286.1.5 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.286.1.6 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

6.286.1.7 virtual decaf::lang::exceptions::InvalidStateException::~InvalidStateException ( ) throw () [inline, virtual]

6.286.2 Member Function Documentation

6.286.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InvalidStateException.h**

6.287 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 ()

6.287.1 Constructor & Destructor Documentation

6.287.1.1 decaf::io::IOException::IOException ( ) throw () [inline]

Default Constructor.

6.287.1.2 decaf::io::IOException::IOException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.287.1.3 decaf::io::IOException::IOException (const IOException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.287.1.4 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

6.287.1.5 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.

6.287.1.6 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

6.287.1.7 virtual decaf::io::IOException::~IOException ( ) throw () [inline, virtual]

6.287.2 Member Function Documentation

6.287.2.1 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. 992).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 1574), **decaf::net::BindException** (p. 413), **decaf::net::ConnectException** (p. 724), **decaf::io::UTFDataFormatException** (p. 2230), **decaf::io::UnsupportedEncodingException** (p. 2187), **decaf::net::HttpRetryException** (p. 1092), **decaf::net::MalformedURLException** (p. 1371), **decaf::net::NoRouteToHostException** (p. 1534), **decaf::net::PortUnreachableException** (p. 1634), **decaf::net::ProtocolException** (p. 1715), **decaf::net::SocketTimeoutException** (p. 1934), **decaf::net::UnknownHostException** (p. 2183), **decaf::net::UnknownServiceException** (p. 2185), **decaf::util::zip::ZipException** (p. 2292), **decaf::io::EOFException** (p. 988), **decaf::io::InterruptedIOException** (p. 1189), and **decaf::net::SocketException** (p. 1916).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**IOException.h**

6.288 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 2161) 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. 2161) and assign its WireFormat instance at creation time.*
- virtual ~**IOTransport** ()
- virtual void **setInputStream** (**decaf::io::DataInputStream** *is)

  *Sets the stream from which this **Transport** (p. 2161) implementation will read its data.*
- virtual void **setOutputStream** (**decaf::io::DataOutputStream** *os)

  *Sets the stream to which this **Transport** (p. 2161) implementation will write its data.*
- virtual void **oneway** (**const Pointer**< **Command** > &command)

  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command)

  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.*

- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command, unsigned int timeout)

  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.*
UnsupportedOperation-Exception	*if this method is not implemented by this transport.*

- virtual **Pointer** < **wireformat::WireFormat** > **getWireFormat** () **const**

  Gets the WireFormat instance that is in use by this transport.
- virtual void **setWireFormat** (**const Pointer**< **wireformat::WireFormat** > &wireFormat)

  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 events from this transport.
- virtual void **start** ()

  *Starts the **Transport*** (p. *2161), the send methods of a **Transport*** (p. *2161) will throw an exception if used before the **Transport*** (p. *2161) is started.*
- virtual void **stop** ()

  *Stops the **Transport*** (p. *2161).*
- virtual void **close** ()

  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. *2161) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () **const**

  *Is this **Transport*** (p. *2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () **const**

  *Is the **Transport*** (p. *2161) Connected to its Broker.*
- virtual bool **isClosed** () **const**

  *Has the **Transport*** (p. *2161) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () **const**
- virtual bool **isReconnectSupported** () **const**
- virtual bool **isUpdateURIsSupported** () **const**
- virtual void **updateURIs** (bool rebalance AMQCPP_UNUSED, **const decaf::util::List**< **decaf::net::URI** > &uris AMQCPP_UNUSED)
- virtual void **reconnect** (**const decaf::net::URI** &uri AMQCPP_UNUSED)

- virtual void **run** ()

  Runs the polling thread.

6.288.1 Detailed Description

Implementation of the **Transport** (p. 2161) 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.

6.288.2 Constructor & Destructor Documentation

6.288.2.1 activemq::transport::IOTransport::IOTransport ( )

Default Constructor.

6.288.2.2 activemq::transport::IOTransport::IOTransport (const Pointer< wireformat::WireFormat > & *wireFormat*)

Create an instance of this **Transport** (p. 2161) and assign its WireFormat instance at creation time.

Parameters

wireFormat	Data encoder / decoder to use when reading and writing.

6.288.2.3 virtual activemq::transport::IOTransport::~IOTransport ( ) [virtual]

6.288.3 Member Function Documentation

6.288.3.1 virtual void activemq::transport::IOTransport::close ( ) [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. 633).

6.288.3.2 virtual std::string activemq::transport::IOTransport::getRemoteAddress ( ) const [inline, virtual]

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2162).

6.288.3.3 virtual TransportListener* activemq::transport::IOTransport::getTransportListener ( ) const [inline, virtual]

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2163).

6.288.3.4 virtual Pointer<wireformat::WireFormat> activemq::transport::IOTransport::getWireFormat ( ) const [inline, virtual]

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a WireFormat info instance.

Returns

The WireFormat the object used to encode / decode commands.

Implements **activemq::transport::Transport** (p. 2163).

6.288.3.5 virtual bool activemq::transport::IOTransport::isClosed ( ) const [inline, virtual]

Has the **Transport** (p. 2161) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2161)

Implements **activemq::transport::Transport** (p. 2163).

6.288.3.6 virtual bool activemq::transport::IOTransport::isConnected ( ) const [inline, virtual]

Is the **Transport** (p. 2161) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2163).

6.288.3.7 virtual bool activemq::transport::IOTransport::isFaultTolerant ( ) const [inline, virtual]

Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2161) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2163).

6.288.3.8 virtual bool activemq::transport::IOTransport::isReconnectSupported ( ) const [inline, virtual]

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.288.3.9 virtual bool activemq::transport::IOTransport::isUpdateURIsSupported ( ) const [inline, virtual]

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.288.3.10 virtual Transport* activemq::transport::IOTransport::narrow (const std::type_info & *typeId*) [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2161) 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. 2164).

6.288.3.11 virtual void activemq::transport::IOTransport::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2164).

6.288.3.12 virtual void activemq::transport::IOTransport::reconnect (const decaf::net::URI &uri *AMQCPP_UNUSED*) [inline, virtual]

This method does nothing in this subclass.

6.288.3.13 virtual Pointer<Response> activemq::transport::IOTransport::request (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

This method always thrown an UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 2165).

6.288.3.14 virtual Pointer<Response> activemq::transport::IOTransport::request (const Pointer< Command > & *command,* unsigned int *timeout*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

This method always thrown an UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 2165).

6.288.3.15 virtual void activemq::transport::IOTransport::run ( ) [virtual]

Runs the polling thread.

Implements **decaf::lang::Runnable** (p. 1793).

6.288.3.16 virtual void activemq::transport::IOTransport::setInputStream (decaf::io::DataInputStream * *is*) [inline, virtual]

Sets the stream from which this **Transport** (p. 2161) implementation will read its data.

Parameters

is	The InputStream that will be read from by this object.

6.288.3.17 virtual void activemq::transport::IOTransport::setOutputStream (decaf::io::DataOutputStream * *os*) [inline, virtual]

Sets the stream to which this **Transport** (p. 2161) implementation will write its data.

Parameters

os	The OuputStream that will be written to by this object.

6.288.3.18 virtual void activemq::transport::IOTransport::setTransportListener (TransportListener * *listener*) [inline, virtual]

Sets the observer of asynchronous events from this transport.

Parameters

listener	the listener of transport events.

Implements **activemq::transport::Transport** (p. 2166).

6.288.3.19 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.

Implements **activemq::transport::Transport** (p. 2166).

6.288.3.20 virtual void activemq::transport::IOTransport::start ( ) [virtual]

Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.

Exceptions

IOException	if and error occurs while starting the **Transport** (p. 2161).

Implements **activemq::transport::Transport** (p. 2166).

6.288.3.21 virtual void activemq::transport::IOTransport::stop ( ) [virtual]

Stops the **Transport** (p. 2161).

Exceptions

IOException	if an error occurs while stopping the transport.

Implements **activemq::transport::Transport** (p. 2167).

6.288.3.22 virtual void activemq::transport::IOTransport::updateURIs (bool rebalance *AMQCPP_UNUSED*, const decaf::util::List< decaf::net::URI > &uris *AMQCPP_UNUSED*) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**IOTransport.h**

6.289 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 1207) 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

6.289.1 Detailed Description

template<typename E>class decaf::lang::Iterable< E >

Implementing this interface allows an object to be cast to an **Iterable** (p. 1207) type for generic collections API calls.

6.289.2 Constructor & Destructor Documentation

6.289.2.1 template<typename E> virtual decaf::lang::Iterable< E >::~Iterable ( ) [inline, virtual]

6.289.3 Member Function Documentation

6.289.3.1 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.

Implemented in **decaf::util::concurrent::LinkedBlockingQueue**< **E** > (p. 1261), **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 806), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 806), **decaf::util::AbstractList**< **E** > (p. 101), **decaf::util::AbstractList**< **Pointer**< **Transport** > > (p. 101), **decaf::util::AbstractList**< **cms::MessageConsumer** * > (p. 101), **decaf::util::AbstractList**< **CompositeTask** * > (p. 101), **decaf::util::AbstractList**< **URI** > (p. 101), **decaf::util::AbstractList**< **Pointer**< **MessageDispatch** > > (p. 101), **decaf::util::AbstractList**< **Pointer**< **DestinationInfo** > > (p. 101), **decaf::util::AbstractList**< **PrimitiveValueNode** > (p. 101), **decaf::util::AbstractList**< **Pointer**< **Command** > > (p. 101), **decaf::util::AbstractList**< **Pointer**< **BackupTransport** > > (p. 101), **decaf::util::AbstractList**< **cms::MessageProducer** * > (p. 101), **decaf::util::AbstractList**< **cms::Destination** * > (p. 101), **decaf::util::AbstractList**< **cms::Session** * > (p. 101), **decaf::util::AbstractList**< **cms::Connection** * > (p. 101), **decaf::util::StlList**< **E** > (p. 1974), **decaf::util::PriorityQueue**< **E** > (p. 1680), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2061), **decaf::util::StlSet**< **E** > (p. 1999), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 1999), **decaf::util::StlSet**< **Resource** * > (p. 1999), **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 820), **decaf::util::AbstractSequentialList**< **E** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **Transport** > > (p. 116), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** * > (p. 116), **decaf::util::AbstractSequentialList**< **CompositeTask** * > (p. 116), **decaf::util::AbstractSequentialList**< **URI** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 116), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 116), **decaf::util::AbstractSequentialList**<

cms::MessageProducer * > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Destination** * > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Session** * > (p. 116), and **decaf::util::AbstractSequentialList**< **cms::Connection** * > (p. 116).

Referenced by decaf::util::AbstractSequentialList< cms::Connection * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::AbstractCollection< cms::Connection * >::addAll(), decaf::util::AbstractList< cms::Connection * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::ArrayList< E >::ArrayList(), decaf::util::AbstractCollection< cms::Connection * >::clear(), decaf::util::AbstractCollection< cms::Connection * >::contains(), decaf::util::AbstractCollection< cms::Connection * >::containsAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::containsAll(), decaf::util::AbstractCollection< cms::Connection * >::copy(), decaf::util::concurrent::CopyOnWriteArraySet< E >::equals(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::equals(), decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue(), decaf::util::AbstractCollection< cms::Connection * >::operator=(), decaf::util::AbstractCollection< cms::Connection * >::remove(), decaf::util::AbstractSet< Resource * >::removeAll(), decaf::util::AbstractCollection< cms::Connection * >::removeAll(), decaf::util::AbstractCollection< cms::Connection * >::retainAll(), and decaf::util::AbstractCollection< cms::Connection * >::toArray().

6.289.3.2 template<typename E> virtual decaf::util::Iterator<E>* decaf::lang::Iterable< E >::iterator ( ) const [pure virtual]

Implemented in **decaf::util::concurrent::LinkedBlockingQueue**< **E** > (p. 1262), **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 806), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 806), **decaf::util::AbstractList**< **E** > (p. 101), **decaf::util::AbstractList**< **Pointer**< **Transport** > > (p. 101), **decaf::util::AbstractList**< **cms::MessageConsumer** * > (p. 101), **decaf::util::AbstractList**< **CompositeTask** * > (p. 101), **decaf::util::AbstractList**< **URI** > (p. 101), **decaf::util::AbstractList**< **Pointer**< **MessageDispatch** > > (p. 101), **decaf::util::AbstractList**< **Pointer**< **DestinationInfo** > > (p. 101), **decaf::util::AbstractList**< **PrimitiveValueNode** > (p. 101), **decaf::util::AbstractList**< **Pointer**< **Command** > > (p. 101), **decaf::util::AbstractList**< **Pointer**< **BackupTransport** > > (p. 101), **decaf::util::AbstractList**< **cms::MessageProducer** * > (p. 101), **decaf::util::AbstractList**< **cms::Destination** * > (p. 101), **decaf::util::AbstractList**< **cms::Session** * > (p. 101), **decaf::util::AbstractList**< **cms::Connection** * > (p. 101), **decaf::util::StlList**< **E** > (p. 1974), **decaf::util::PriorityQueue**< **E** > (p. 1680), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 2061), **decaf::util::StlSet**< **E** > (p. 2000), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 2000), **decaf::util::StlSet**< **Resource** * > (p. 2000), **decaf::util::concurrent::CopyOnWriteArraySet**< **E** > (p. 820), **decaf::util::AbstractSequentialList**< **E** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **Transport** > > (p. 116), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** * > (p. 116), **decaf::util::AbstractSequentialList**< **CompositeTask** * > (p. 116), **decaf::util::AbstractSequentialList**< **URI** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 116), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 116), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** * > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Destination** * > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Session** * > (p. 116), and **decaf::util::AbstractSequentialList**< **cms::Connection** * > (p. 116).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Iterable.h**

6.290 decaf::util::Iterator< E > 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< E >:

Public Member Functions

- virtual ~**Iterator** ()
- virtual E **next** ()=0

  Returns the next element in the iteration.
- virtual bool **hasNext** () **const** =0

  Returns true if the iteration has more elements.
- virtual void **remove** ()=0

  Removes from the underlying collection the last element returned by the iterator (optional operation).

6.290.1 Detailed Description

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

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.

6.290.2 Constructor & Destructor Documentation

6.290.2.1 template<typename E> virtual decaf::util::Iterator< E >::~Iterator ( ) [inline, virtual]

6.290.3 Member Function Documentation

6.290.3.1 template<typename E> virtual bool decaf::util::Iterator< E >::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** (p. 1537) to be thrown.

Returns

true if there are more elements available for iteration.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 356).

6.290.3.2 template<typename E> virtual E decaf::util::Iterator< E >::next ( ) [pure virtual]

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p. 1209) method returns false will return each element in the underlying collection exactly once.

Returns

the next element in the iteration of elements.

Exceptions

NoSuchElementException *(p. 1537)*	if the iteration has no more elements.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 357).

6.290.3.3 template<typename E> virtual void decaf::util::Iterator< E >::remove ( ) [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

UnsupportedOperation-Exception	if the remove operation is not supported by this **Iterator** (p. 1209).
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.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 358).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Iterator.h**

6.291 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **JournalQueueAck** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 Attributes

- **Pointer< ActiveMQDestination > destination**
- **Pointer< MessageAck > messageAck**

6.291.1 Constructor & Destructor Documentation

6.291.1.1 activemq::commands::JournalQueueAck::JournalQueueAck ( )

6.291.1.2 virtual activemq::commands::JournalQueueAck::~JournalQueueAck ( ) [virtual]

6.291.2 Member Function Documentation

6.291.2.1 virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.291.2.2 virtual void activemq::commands::JournalQueueAck::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. 879).

6.291.2.3 virtual bool activemq::commands::JournalQueueAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.291.2.4 virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.291.2.5 virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ( ) const [virtual]

6.291.2.6 virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ( ) [virtual]

6.291.2.7 virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ( ) const [virtual]

6.291.2.8 virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ( ) [virtual]

6.291.2.9 virtual void activemq::commands::JournalQueueAck::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.291.2.10 virtual void activemq::commands::JournalQueueAck::setMessageAck (const Pointer< MessageAck > & *messageAck*) [virtual]

6.291.2.11 virtual std::string activemq::commands::JournalQueueAck::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.291.3 Field Documentation

6.291.3.1 Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination [protected]

6.291.3.2 const unsigned char activemq::commands::JournalQueueAck::ID_JOURNALQUEUEACK = 52 [static]

6.291.3.3 Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalQueueAck.h**

6.292 activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1213).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/JournalQueueAckMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual ~**JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.292.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1213).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.292.2 Constructor & Destructor Documentation

6.292.2.1 activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::JournalQueueAckMarshaller ( ) [inline]

6.292.2.2 virtual activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::~JournalQueueAckMarshaller ( ) [inline, virtual]

6.292.3 Member Function Documentation

6.292.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.292.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.292.3.3 virtual void activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.292.3.4 virtual void activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.292.3.5 virtual int activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.292.3.6 virtual void activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.292.3.7 virtual void activemq::wireformat::openwire::marshal::generated::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalQueueAckMarshaller.h**

6.293 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **JournalTopicAck** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 & **getSubscritionName** () **const**
- virtual std::string & **getSubscritionName** ()
- virtual void **setSubscritionName** (**const** std::string &**subscritionName**)
- 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 Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageId** > **messageId**
- long long **messageSequenceId**
- std::string **subscritionName**
- std::string **clientId**
- **Pointer**< **TransactionId** > **transactionId**

6.293.1 Constructor & Destructor Documentation

6.293.1.1 activemq::commands::JournalTopicAck::JournalTopicAck ( )

6.293.1.2 virtual activemq::commands::JournalTopicAck::~JournalTopicAck ( ) [virtual]

6.293.2 Member Function Documentation

6.293.2.1 virtual JournalTopicAck* activemq::commands::JournalTopicAck::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.293.2.2 virtual void activemq::commands::JournalTopicAck::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. 879).

6.293.2.3 virtual bool activemq::commands::JournalTopicAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.293.2.4 virtual const std::string& activemq::commands::JournalTopicAck::getClientId ( ) const [virtual]

6.293.2.5 virtual std::string& activemq::commands::JournalTopicAck::getClientId ( ) [virtual]

6.293.2.6 virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.293.2.7 virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination ( ) const [virtual]

6.293.2.8 virtual Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination ( ) [virtual]

6.293.2.9 virtual const Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId ( ) const [virtual]

6.293.2.10 virtual Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId ( ) [virtual]

6.293.2.11 virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId ( ) const [virtual]

6.293.2.12 virtual const std::string& activemq::commands::JournalTopicAck::getSubscritionName ( ) const [virtual]

6.293.2.13 virtual std::string& activemq::commands::JournalTopicAck::getSubscritionName ( ) [virtual]

6.293.2.14 virtual const Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId ( ) const [virtual]

6.293.2.15 virtual Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId ( ) [virtual]

6.293.2.16 virtual void activemq::commands::JournalTopicAck::setClientId (const std::string & *clientId*) [virtual]

6.293.2.17 virtual void activemq::commands::JournalTopicAck::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.293.2.18 virtual void activemq::commands::JournalTopicAck::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]

6.293.2.19 virtual void activemq::commands::JournalTopicAck::setMessageSequenceId (long long *messageSequenceId*) [virtual]

6.293.2.20 virtual void activemq::commands::JournalTopicAck::setSubscritionName (const std::string & *subscritionName*) [virtual]

6.293.2.21 virtual void activemq::commands::JournalTopicAck::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.293.2.22 virtual std::string activemq::commands::JournalTopicAck::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.293.3 Field Documentation

6.293.3.1 std::string activemq::commands::JournalTopicAck::clientId [protected]

6.293.3.2 Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination [protected]

6.293.3.3 const unsigned char activemq::commands::JournalTopicAck::ID_JOURNALTOPICACK = 50 [static]

6.293.3.4 Pointer<MessageId> activemq::commands::JournalTopicAck::messageId [protected]

6.293.3.5 long long activemq::commands::JournalTopicAck::messageSequenceId [protected]

6.293.3.6 std::string activemq::commands::JournalTopicAck::subscritionName [protected]

6.293.3.7 Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTopicAck.h**

6.294 activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1219).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/JournalTopicAckMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual ~**JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.294.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1219).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.294.2 Constructor & Destructor Documentation

6.294.2.1 activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]

6.294.2.2 virtual activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::~JournalTopicAckMarshaller ( ) [inline, virtual]

6.294.3 Member Function Documentation

6.294.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.294.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.294.3.3 virtual void activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.294.3.4 virtual void activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.294.3.5 virtual int activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.294.3.6 virtual void activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.294.3.7 virtual void activemq::wireformat::openwire::marshal::generated::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalTopicAckMarshaller.h**

6.295 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **JournalTrace** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 Attributes

- std::string **message**

6.295.1 Constructor & Destructor Documentation

6.295.1.1 activemq::commands::JournalTrace::JournalTrace ( )

6.295.1.2 virtual activemq::commands::JournalTrace::~JournalTrace ( ) [virtual]

6.295.2 Member Function Documentation

6.295.2.1 virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.295.2.2 virtual void activemq::commands::JournalTrace::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. 879).

6.295.2.3 virtual bool activemq::commands::JournalTrace::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.295.2.4 virtual unsigned char activemq::commands::JournalTrace::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.295.2.5 virtual const std::string& activemq::commands::JournalTrace::getMessage ( ) const [virtual]

6.295.2.6 virtual std::string& activemq::commands::JournalTrace::getMessage ( ) [virtual]

6.295.2.7 virtual void activemq::commands::JournalTrace::setMessage (const std::string & *message*) [virtual]

6.295.2.8 virtual std::string activemq::commands::JournalTrace::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.295.3 Field Documentation

6.295.3.1 const unsigned char activemq::commands::JournalTrace::ID_JOURNALTRACE = 53 [static]

6.295.3.2 std::string activemq::commands::JournalTrace::message [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTrace.h**

6.296 activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1224).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/JournalTraceMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.296.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1224).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.296.2 Constructor & Destructor Documentation

6.296.2.1 activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::JournalTraceMarshaller ( ) [inline]

6.296.2.2 virtual activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::∼JournalTraceMarshaller ( ) [inline, virtual]

6.296.3 Member Function Documentation

6.296.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.296.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.296.3.3 virtual void activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::looseMarshal (OpenWireFormat ∗ *format,* commands::DataStructure ∗ *command,* decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.296.3.4 virtual void activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.296.3.5 virtual int activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.296.3.6 virtual void activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.296.3.7 virtual void activemq::wireformat::openwire::marshal::generated::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalTraceMarshaller.h**

6.297 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **JournalTransaction** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*

- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 Attributes

- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**
- bool **wasPrepared**

6.297.1 Constructor & Destructor Documentation

6.297.1.1 activemq::commands::JournalTransaction::JournalTransaction ( )

6.297.1.2 virtual activemq::commands::JournalTransaction::~JournalTransaction ( ) [virtual]

6.297.2 Member Function Documentation

6.297.2.1 virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.297.2.2 virtual void activemq::commands::JournalTransaction::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. 879).

6.297.2.3 virtual bool activemq::commands::JournalTransaction::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.297.2.4 virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.297.2.5 virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId ( ) const [virtual]

6.297.2.6 virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId ( ) [virtual]

6.297.2.7 virtual unsigned char activemq::commands::JournalTransaction::getType ( ) const [virtual]

6.297.2.8 virtual bool activemq::commands::JournalTransaction::getWasPrepared ( ) const [virtual]

6.297.2.9 virtual void activemq::commands::JournalTransaction::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.297.2.10 virtual void activemq::commands::JournalTransaction::setType (unsigned char *type*) [virtual]

6.297.2.11 virtual void activemq::commands::JournalTransaction::setWasPrepared (bool *wasPrepared*) [virtual]

6.297.2.12 virtual std::string activemq::commands::JournalTransaction::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.297.3 Field Documentation

6.297.3.1 const unsigned char activemq::commands::JournalTransaction::ID_JOURNALTRANSACTION = 54 [static]

6.297.3.2 Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId [protected]

6.297.3.3 unsigned char activemq::commands::JournalTransaction::type [protected]

6.297.3.4 bool activemq::commands::JournalTransaction::wasPrepared [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTransaction.h**

6.298 activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller Class Reference

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

`#include <src/main/activemq/wireformat/openwire/marshal/generated/JournalTransactionMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.298.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (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

6.298.2 Constructor & Destructor Documentation

6.298.2.1 activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::JournalTransactionMarshaller ( ) [inline]

6.298.2.2 virtual activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::~JournalTransactionMarshaller ( ) [inline, virtual]

6.298.3 Member Function Documentation

6.298.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.298.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.298.3.3 virtual void activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.298.3.4 virtual void activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.298.3.5 virtual int activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.298.3.6 virtual void activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.298.3.7 virtual void activemq::wireformat::openwire::marshal::generated::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *ds*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**JournalTransactionMarshaller.h**

6.299 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **KeepAliveInfo** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () **const**
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ∗visitor)

  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_KEEPALIVEINFO** = 10

6.299.1 Constructor & Destructor Documentation

6.299.1.1 activemq::commands::KeepAliveInfo::KeepAliveInfo ( )

6.299.1.2 virtual activemq::commands::KeepAliveInfo::∼KeepAliveInfo ( ) [virtual]

6.299.2 Member Function Documentation

6.299.2.1 virtual KeepAliveInfo* activemq::commands::KeepAliveInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.299.2.2 virtual void activemq::commands::KeepAliveInfo::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. 382).

6.299.2.3 virtual bool activemq::commands::KeepAliveInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.299.2.4 virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.299.2.5 virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isKeepAliveInfo()** (p. 1235) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.299.2.6 virtual std::string activemq::commands::KeepAliveInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.299.2.7 virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.299.3 Field Documentation

6.299.3.1 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**

6.300 activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1236).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/KeepAliveInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual ~**KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.300.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1236).

NOTE!: This file is 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::generated::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]

6.300.2.2 virtual activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::∼KeepAliveInfoMarshaller ( ) [inline, virtual]

6.300.3 Member Function Documentation

6.300.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.300.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.300.3.3 virtual void activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.300.3.4 virtual void activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.300.3.5 virtual int activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.300.3.6 virtual void activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.300.3.7 virtual void activemq::wireformat::openwire::marshal::generated::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**KeepAliveInfoMarshaller.h**

6.301 decaf::security::Key Class Reference

The **Key** (p. 1239) 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.

6.301.1 Detailed Description

The **Key** (p. 1239) 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. 1239) 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.

6.301.2 Constructor & Destructor Documentation

6.301.2.1 virtual decaf::security::Key::~Key ( ) [inline, virtual]

6.301.3 Member Function Documentation

6.301.3.1 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.

6.301.3.2 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.

6.301.3.3 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**

6.302 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 decaf::lang::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 ()

6.302.1 Constructor & Destructor Documentation

6.302.1.1 decaf::security::KeyException::KeyException ( ) throw () [inline]

Default Constructor.

6.302.1.2 decaf::security::KeyException::KeyException (const decaf::lang::Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception

6.302.1.3 decaf::security::KeyException::KeyException (const KeyException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.302.1.4 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

6.302.1.5 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.

6.302.1.6 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

6.302.1.7 virtual decaf::security::KeyException::~KeyException ( ) throw () [inline, virtual]

6.302.2 Member Function Documentation

6.302.2.1 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. 1081).

Reimplemented in **decaf::security::InvalidKeyException** (p. 1193), and **decaf::security::KeyManagementException** (p. 1245).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**KeyException.h**

6.303 decaf::security::KeyManagementException Class Reference

`#include <src/main/decaf/security/KeyManagementException.h>`

Inheritance diagram for decaf::security::KeyManagementException:

Public Member Functions

- **KeyManagementException** () throw ()
  Default Constructor.
- **KeyManagementException** (**const Exception** &ex) throw ()
  Conversion Constructor from some other Exception.
- **KeyManagementException** (**const KeyManagementException** &ex) throw ()
  Copy Constructor.
- **KeyManagementException** (**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.
- **KeyManagementException** (**const** std::exception *__cause__) throw ()
  Constructor.
- **KeyManagementException** (**const** char *file, **const** int lineNumber, **const** char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **KeyManagementException** * **clone** () **const**
  Clones this exception.
- virtual ~**KeyManagementException** () throw ()

6.303.1 Constructor & Destructor Documentation

6.303.1.1 decaf::security::KeyManagementException::KeyManagementException ( ) throw () [inline]

Default Constructor.

6.303.1.2 decaf::security::KeyManagementException::KeyManagementException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	An exception that should become this type of Exception

6.303.1.3 decaf::security::KeyManagementException::KeyManagementException (const KeyManagementException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.303.1.4 decaf::security::KeyManagementException::KeyManagementException (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

6.303.1.5 decaf::security::KeyManagementException::KeyManagementException (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.

6.303.1.6 decaf::security::KeyManagementException::KeyManagementException (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

6.303.1.7 virtual decaf::security::KeyManagementException::~KeyManagementException ( ) throw ()

`[inline, virtual]`

6.303.2 Member Function Documentation

6.303.2.1 virtual KeyManagementException* decaf::security::KeyManagementException::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. 1242).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**KeyManagementException.h**

6.304 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **LastPartialCommand** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*

Static Public Attributes

- static **const** unsigned char **ID_LASTPARTIALCOMMAND** = 61

6.304.1 Constructor & Destructor Documentation

6.304.1.1 activemq::commands::LastPartialCommand::LastPartialCommand ( )

6.304.1.2 virtual activemq::commands::LastPartialCommand::~LastPartialCommand ( ) [virtual]

6.304.2 Member Function Documentation

6.304.2.1 virtual LastPartialCommand* activemq::commands::LastPartialCommand::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 1609).

6.304.2.2 virtual void activemq::commands::LastPartialCommand::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::PartialCommand** (p. 1609).

6.304.2.3 virtual bool activemq::commands::LastPartialCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::PartialCommand** (p. 1609).

6.304.2.4 virtual unsigned char activemq::commands::LastPartialCommand::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::PartialCommand** (p. 1610).

6.304.2.5 virtual std::string activemq::commands::LastPartialCommand::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 1610).

6.304.3 Field Documentation

6.304.3.1 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**

6.305 activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1247).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/LastPartialCommandMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual ~**LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.305.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1247).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.305.2 Constructor & Destructor Documentation

6.305.2.1 activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::LastPartialCommandMarshaller ( ) [inline]

6.305.2.2 virtual activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::~LastPartialCommandMarshaller ( ) [inline, virtual]

6.305.3 Member Function Documentation

6.305.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612).

6.305.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612).

6.305.3.3 virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612).

6.305.3.4 virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1612).

6.305.3.5 virtual int activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1613).

6.305.3.6 virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1613).

6.305.3.7 virtual void activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller** (p. 1614).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**LastPartialCommandMarshaller.h**

6.306 decaf::util::comparators::Less< E > Class Template Reference

Simple **Less** (p. 1250) **Comparator** (p. 689) 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. 689) to be passed to an STL **Map** (p. 1371) for use as the sorting criteria.*
- virtual int **compare** (**const** E &o1, **const** E &o2) **const**

  Compares its two arguments for order.

6.306.1 Detailed Description

template<typename E>class decaf::util::comparators::Less< E >

Simple **Less** (p. 1250) **Comparator** (p. 689) that compares to elements to determine if the first is less than the second.

This can be used in **Collection** (p. 660) classes to sort elements according to their natural ordering. By design the **Comparator** (p. 689)'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

6.306.2 Constructor & Destructor Documentation

6.306.2.1 template<typename E > decaf::util::comparators::Less< E >::Less ( ) [inline]

6.306.2.2 template<typename E > virtual decaf::util::comparators::Less< E >::~Less ( ) [inline, virtual]

6.306.3 Member Function Documentation

6.306.3.1 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 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.

Implements **decaf::util::Comparator**< E > (p. 689).

6.306.3.2 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. 689) to be passed to an STL **Map** (p. 1371) 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. 690).

The documentation for this class was generated from the following file:

- src/main/decaf/util/comparators/**Less.h**

6.307 std::less< decaf::lang::ArrayPointer< 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/ArrayPointer.h>`

Public Member Functions

- bool **operator()** (**const decaf::lang::ArrayPointer**< T > &left, **const decaf::lang::ArrayPointer**< T > &right) **const**

6.307.1 Detailed Description

template<typename T>struct std::less< decaf::lang::ArrayPointer< T > >

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.307.2 Member Function Documentation

6.307.2.1 template<typename T > bool std::less< decaf::lang::ArrayPointer< T > >::operator() (const decaf::lang::ArrayPointer< T > & *left,* const decaf::lang::ArrayPointer< T > & *right*) const [inline]

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

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

6.308 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>`

Public Member Functions

- bool **operator()** (**const decaf::lang::Pointer**< T > &left, **const decaf::lang::Pointer**< T > &right) **const**

6.308.1 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.

6.308.2 Member Function Documentation

6.308.2.1 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**

6.309 decaf::util::logging::Level Class Reference

The **Level** (p. 1253) class defines a set of standard logging levels that can be used to control logging output.

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

Inheritance diagram for decaf::util::logging::Level:

Public Member Functions

- virtual ~**Level** ()
- int **intValue** () **const**
- std::string **getName** () **const**
- std::string **toString** () **const**
- virtual int **compareTo** (**const Level** &value) **const**
- virtual bool **equals** (**const Level** &value) **const**
- virtual bool **operator==** (**const Level** &value) **const**
- virtual bool **operator<** (**const Level** &value) **const**

Static Public Member Functions

- static **Level parse** (**const** std::string &name)

  *Parse a level name string into a **Level** (p. 1253).*

Static Public Attributes

- static **const Level INHERIT**

  *NULL is a special level that indicates that the **Logger** (p. 1312) should get its **Level** (p. 1253) from its parent **Logger** (p. 1312), the value is initialized as zero.*
- static **const Level OFF**

  OFF is a special level that can be used to turn off logging.
- static **const Level SEVERE**

  SEVERE is a message level indicating a serious failure.
- static **const Level WARNING**

WARNING is a message level indicating a potential problem.

- static **const Level INFO**

  INFO is a message level for informational messages.

- static **const Level DEBUG**

  DEBUG is a level for more verbose informative messages.

- static **const Level CONFIG**

  CONFIG is a message level for static configuration messages.

- static **const Level FINE**

  FINE is a message level providing tracing information.

- static **const Level FINER**

  FINER indicates a fairly detailed tracing message.

- static **const Level FINEST**

  FINEST indicates a highly detailed tracing message.

- static **const Level ALL**

  ALL indicates that all messages should be logged.

Protected Member Functions

- **Level** (**const** std::string &name, int value)

  *Create a named **Level** (p. 1253) with a given integer value.*

6.309.1 Detailed Description

The **Level** (p. 1253) class defines a set of standard logging levels that can be used to control logging output.

The logging **Level** (p. 1253) objects are ordered and are specified by ordered integers. Enabling logging at a given level also enables logging at all higher levels.

Clients should normally use the predefined **Level** (p. 1253) constants such as **Level.SEVERE** (p. 1257).

The levels in descending order are:

- SEVERE (highest value)
  - WARNING
  - INFO
  - DEBUG
  - CONFIG
  - FINE
  - FINER
  - FINEST (lowest value)

In addition there is a level OFF that can be used to turn off logging, and a level ALL that can be used to enable logging of all messages.

It is possible for third parties to define additional logging levels by subclassing **Level** (p. 1253). In such cases subclasses should take care to chose unique integer level values.

Since

1.0

6.309.2 Constructor & Destructor Documentation

6.309.2.1 decaf::util::logging::Level::Level (const std::string & *name,* int *value*) [protected]

Create a named **Level** (p. 1253) with a given integer value.

Parameters

name	Name of the level, e.g. SEVERE
value	Unique integer value of this level, e.g. 100

6.309.2.2 virtual decaf::util::logging::Level::~Level ( ) [virtual]

6.309.3 Member Function Documentation

6.309.3.1 virtual int decaf::util::logging::Level::compareTo (const Level & *value*) const [virtual]

6.309.3.2 virtual bool decaf::util::logging::Level::equals (const Level & *value*) const [virtual]

6.309.3.3 std::string decaf::util::logging::Level::getName ( ) const [inline]

Returns

the name of this **Level** (p. 1253) instance.

6.309.3.4 int decaf::util::logging::Level::intValue ( ) const [inline]

Returns

the integer value of this level instance.

6.309.3.5 virtual bool decaf::util::logging::Level::operator< (const Level & *value*) const [virtual]

6.309.3.6 virtual bool decaf::util::logging::Level::operator== (const Level & *value*) const [virtual]

6.309.3.7 static Level decaf::util::logging::Level::parse (const std::string & *name*) [static]

Parse a level name string into a **Level** (p. 1253).

The argument string may consist of either a level name or an integer value.

For example:

- "SEVERE"
  - "1000"

Parameters

name	- The name or int value of the desired **Level** (p. 1253)

Returns

the parsed **Level** (p. 1253) value, passing in a level name that is an int value that is not one of the known **Level** (p. 1253) values will result in a new **Level** (p. 1253) that has been initialized with that int value and name as the string form of the int.

Exceptions

IllegalArgumentException	if the value is not valid, validity means that the string is either a valid int (between Integer-::MIN_VALUE and Integer::MAX_VALUE or is one of the known level names.

6.309.3.8 std::string decaf::util::logging::Level::toString ( ) const [inline]

Returns

the string value of this **Level** (p. 1253), e.g. "SEVERE".

6.309.4 Field Documentation

6.309.4.1 const Level decaf::util::logging::Level::ALL [static]

ALL indicates that all messages should be logged.

This level is initialized to Integer::MIN_VALUE.

6.309.4.2 const Level decaf::util::logging::Level::CONFIG [static]

CONFIG is a message level for static configuration messages.

CONFIG messages are intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the System properties, etc. This level is initialized to 600.

6.309.4.3 const Level decaf::util::logging::Level::DEBUG [static]

DEBUG is a level for more verbose informative messages.

DEBUG messages are intended to provide a more detailed message intended for use by developers in tracking the behavior of a client. DEBUG messages typically contain more implementation specific information that might not be significant to end users or system admins. This level is initialized to 700.

6.309.4.4 const Level decaf::util::logging::Level::FINE [static]

FINE is a message level providing tracing information.

All of FINE, FINER, and FINEST are intended for relatively detailed tracing. The exact meaning of the three levels will vary between subsystems, but in general, FINEST should be used for the most detailed output, FINER for somewhat less detailed output, and FINE for the lowest volume (and most important) messages.

In general the FINE level should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem.

FINE messages might include things like minor (recoverable) failures. Issues indicating potential performance problems are also worth logging as FINE. This level is initialized to 500.

6.309.4.5 const Level decaf::util::logging::Level::FINER [static]

FINER indicates a fairly detailed tracing message.

By default logging calls for entering, returning, or throwing an exception are traced at this level. This level is initialized to 400.

6.309.4.6 const Level decaf::util::logging::Level::FINEST [static]

FINEST indicates a highly detailed tracing message.

This level is initialized to 300.

6.309.4.7 const Level decaf::util::logging::Level::INFO [static]

INFO is a message level for informational messages.

Typically INFO messages will be written to the console or its equivalent. So the INFO level should only be used for reasonably significant messages that will make sense to end users and system admins. This level is initialized to 800.

6.309.4.8 const Level decaf::util::logging::Level::INHERIT [static]

NULL is a special level that indicates that the **Logger** (p. 1312) should get its **Level** (p. 1253) from its parent **Logger** (p. 1312), the value is initialized as zero.

6.309.4.9 const Level decaf::util::logging::Level::OFF [static]

OFF is a special level that can be used to turn off logging.

This level is initialized to Integer::MAX_VALUE

6.309.4.10 const Level decaf::util::logging::Level::SEVERE [static]

SEVERE is a message level indicating a serious failure.

In general SEVERE messages should describe events that are of considerable importance and which will prevent normal program execution. They should be reasonably intelligible to end users and to system administrators. This level is initialized to 1000.

6.309.4.11 const Level decaf::util::logging::Level::WARNING [static]

WARNING is a message level indicating a potential problem.

In general WARNING messages should describe events that will be of interest to end users or system managers, or which indicate potential problems. This level is initialized to 900.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Level.h**

6.310 decaf::util::concurrent::LinkedBlockingQueue< E > Class Template Reference

A **BlockingQueue** (p. 417) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.

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

Inheritance diagram for decaf::util::concurrent::LinkedBlockingQueue< E >:

Data Structures

- class **ConstLinkedIterator**
- class **LinkedIterator**
- class **QueueNode**
- class **TotalLock**

Public Member Functions

- **LinkedBlockingQueue** ()

  Create a new instance with a Capacity of Integer::MAX_VALUE.
- **LinkedBlockingQueue** (int capacity)

  Create a new instance with the given initial capacity value.
- **LinkedBlockingQueue** (**const Collection**< E > &collection)

  *Create a new instance with a Capacity of Integer::MAX_VALUE and adds all the values contained in the specified collection to this **Queue** (p. 1723).*
- virtual ~**LinkedBlockingQueue** ()
- **LinkedBlockingQueue**< E > & **operator=** (**const LinkedBlockingQueue**< E > &queue)
- **LinkedBlockingQueue**< E > & **operator=** (**const Collection**< E > &collection)
- virtual int **size** () **const**

  Returns the number of elements in this collection.
- virtual void **clear** ()

  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. 1210) 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

UnsupportedOperation-Exception	*if the clear operation is not supported by this collection*

  This implementation repeatedly invokes poll until it returns false.
- virtual int **remainingCapacity** () **const**

  Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.
- virtual void **put** (**const** E &value)

  Inserts the specified element into this queue, waiting if necessary for space to become available.
- virtual bool **offer** (**const** E &value, long long timeout, **const TimeUnit** &unit)

  Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
- virtual bool **offer** (**const** E &value)

  Inserts the specified element into the queue provided that the condition allows such an operation.
- virtual E **take** ()

  Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.
- virtual bool **poll** (E &result, long long timeout, **const TimeUnit** &unit)

  Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.
- 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 bool **remove** (**const** E &value)

Removes a single instance of the specified element from the collection.
More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this* ***Collection*** *(p. 660).*

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the* ***Collection*** *(p. 660) is a container of pointers and does not allow NULL values.*

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.

- virtual std::vector< E > **toArray** () **const**

  Answers an STL vector containing copies of all elements contained in this ***Collection*** *(p. 660).*
- virtual std::string **toString** () **const**
- virtual int **drainTo** (**Collection**< E > &c)

  Removes all available elements from this queue and adds them to the given collection.
- virtual int **drainTo** (**Collection**< E > &sink, int maxElements)

  Removes at most the given number of available elements from this queue and adds them to the given collection.
- virtual **decaf::util::Iterator** < E > * **iterator** ()
- virtual **decaf::util::Iterator** < E > * **iterator** () **const**

6.310.1 Detailed Description

template<typename E>class decaf::util::concurrent::LinkedBlockingQueue< E >

A **BlockingQueue** (p. 417) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.

Elements are inserted and removed in FIFO order. The internal structure of the queue is based on a linked nodes which provides for better performance over their array based versions but the performance is less predictable.

The capacity bound of this class default to Integer::MAX_VALUE.

Since

1.0

6.310.2 Constructor & Destructor Documentation

6.310.2.1 template<typename E> decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue ( ) [inline]

Create a new instance with a Capacity of Integer::MAX_VALUE.

6.310.2.2 template<typename E> decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue (int *capacity*) [inline]

Create a new instance with the given initial capacity value.

Parameters

capacity	The initial capacity value to assign to this **Queue** (p. 1723).

Exceptions

IllegalArgumentException	if the specified capacity is not greater than zero.

6.310.2.3 template<typename E> decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedBlockingQueue (const Collection< E > & *collection*) [inline]

Create a new instance with a Capacity of Integer::MAX_VALUE and adds all the values contained in the specified collection to this **Queue** (p. 1723).

Parameters

collection	The **Collection** (p. 660) whose elements are to be copied to this **Queue** (p. 1723).

Exceptions

IllegalStateException	if the number of elements in the collection exceeds this **Queue** (p. 1723)'s capacity.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::lang::Iterable< E >::iterator().

6.310.2.4 template<typename E> virtual decaf::util::concurrent::LinkedBlockingQueue< E >::~LinkedBlockingQueue ( ) [inline, virtual]

6.310.3 Member Function Documentation

6.310.3.1 template<typename E> virtual void decaf::util::concurrent::LinkedBlockingQueue< E >::clear ( ) [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. 1210) 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

This implementation repeatedly invokes poll until it returns false.

This implementation repeatedly invokes poll until it returns false.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 110).

References decaf::util::concurrent::atomic::AtomicInteger::getAndSet(), decaf::util::AbstractCollection< E >::lock(), decaf::util::concurrent::Mutex::notify(), and decaf::util::concurrent::atomic::AtomicInteger::set().

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::operator=().

6.310.3.2 template<typename E> virtual int decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo (Collection< E > & *c*) [inline, virtual]

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c	the collection to transfer elements into

Returns

the number of elements transferred

Exceptions

UnsupportedOperation-Exception	if addition of elements is not supported by the specified collection
IllegalArgumentException	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 419).

References decaf::lang::Integer::MAX_VALUE.

6.310.3.3 template<typename E> virtual int decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo (Collection< E > & *c*, int *maxElements*) [inline, virtual]

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c	the collection to transfer elements into
maxElements	the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

UnsupportedOperation-Exception	if addition of elements is not supported by the specified collection
IllegalArgumentException	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 420).

References decaf::util::Collection< E >::add(), decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndAdd(), and decaf::lang::Math::min().

6.310.3.4 template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::LinkedBlockingQueue< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1207).

6.310.3.5 template<typename E> virtual decaf::util::Iterator<E>* decaf::util::concurrent::LinkedBlockingQueue< E >::iterator ( ) const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1208).

6.310.3.6 template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< E >::offer (const E & *e,* long long *timeout,* const TimeUnit & *unit*) [inline, virtual]

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Parameters

e	the element to add
timeout	how long to wait before giving up, in units of `unit`
unit	a **`TimeUnit`** (p. 2134) determining how to interpret the `timeout` parameter

Returns

`true` if successful, or `false` if the specified waiting time elapses before space is available

Exceptions

InterruptedException	if interrupted while waiting
NullPointerException	if the specified element is null
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 420).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement(), decaf::util::concurrent::Mutex::notify(), decaf::util::concurrent::TimeUnit::toMillis(), and decaf::util::concurrent::Mutex::wait().

6.310.3.7 template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< E >::offer (const E & *value*) [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. 1723) implementation does not allow Null values to be inserted into the **Queue** (p. 1723).
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 1724).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement(), and decaf::util::concurrent::Mutex::notify().

6.310.3.8 template<typename E> LinkedBlockingQueue<E>& decaf::util::concurrent::LinkedBlockingQueue< E >::operator= (const LinkedBlockingQueue< E > & *queue*) [inline]

References decaf::util::AbstractQueue< E >::addAll(), and decaf::util::concurrent::LinkedBlockingQueue< E >::clear().

6.310.3.9 template<typename E> LinkedBlockingQueue<E>& decaf::util::concurrent::LinkedBlockingQueue< E >::operator= (const Collection< E > & *collection*) [inline]

References decaf::util::AbstractQueue< E >::addAll(), and decaf::util::concurrent::LinkedBlockingQueue< E >::clear().

6.310.3.10 template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< 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. 1725).

References decaf::util::concurrent::atomic::AtomicInteger::get(), and decaf::lang::Pointer< T, REFCOUNTER >::get().

6.310.3.11 template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< E >::poll (E & *result,* long long *timeout,* const TimeUnit & *unit*) [inline, virtual]

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

Parameters

result	the referenced value that will be assigned the value retrieved from the **Queue** (p. 1723). Undefined if this methods returned false.
timeout	how long to wait before giving up, in units of `unit`
unit	a **`TimeUnit`** (p. 2134) determining how to interpret the `timeout` parameter.

Returns

`true` if successful or `false` if the specified waiting time elapses before an element is available.

Exceptions

InterruptedException	if interrupted while waiting

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 421).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement(), decaf::util::concurrent::Mutex::notify(), decaf::util::concurrent::TimeUnit::toMillis(), and decaf::util::concurrent::Mutex::wait().

6.310.3.12 template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< 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. 1723) 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. 1725).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement(), and decaf::util::concurrent::Mutex::notify().

6.310.3.13 template<typename E> virtual void decaf::util::concurrent::LinkedBlockingQueue< E >::put (const E & *value*) [inline, virtual]

Inserts the specified element into this queue, waiting if necessary for space to become available.

Parameters

value	the element to add

Exceptions

InterruptedException	if interrupted while waiting
NullPointerException	if the specified element is null
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 421).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement(), decaf::util::concurrent::Mutex::notify(), and decaf::util::concurrent::Mutex::wait().

6.310.3.14 template<typename E> virtual int decaf::util::concurrent::LinkedBlockingQueue< E >::remainingCapacity ( ) const [inline, virtual]

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

Returns

the remaining capacity

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 421).

References decaf::util::concurrent::atomic::AtomicInteger::get().

6.310.3.15 template<typename E> virtual bool decaf::util::concurrent::LinkedBlockingQueue< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 92).

References decaf::util::AbstractCollection< E >::lock().

6.310.3.16 template<typename E> virtual int decaf::util::concurrent::LinkedBlockingQueue< 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. 669).

References decaf::util::concurrent::atomic::AtomicInteger::get().

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::toArray().

6.310.3.17 template<typename E> virtual E decaf::util::concurrent::LinkedBlockingQueue< E >::take ( ) [inline, virtual]

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Returns

the head of this queue

Exceptions

InterruptedException	if interrupted while waiting

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 422).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement(), decaf::util::concurrent::Mutex::notify(), and decaf::util::concurrent::Mutex::wait().

6.310.3.18 template<typename E> virtual std::vector<E> decaf::util::concurrent::LinkedBlockingQueue< E >::toArray ( ) const [inline, virtual]

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).

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. 660)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 94).

References decaf::util::concurrent::atomic::AtomicInteger::get(), decaf::util::AbstractCollection< E >::lock(), and decaf::util::concurrent::LinkedBlockingQueue< E >::size().

6.310.3.19 template<typename E> virtual std::string decaf::util::concurrent::LinkedBlockingQueue< E >::toString ( ) const [inline, virtual]

References decaf::util::concurrent::atomic::AtomicInteger::get().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**LinkedBlockingQueue.h**

6.311 decaf::util::LinkedList< E > Class Template Reference

A complete implementation of the **List** (p. 1286) interface using a doubly linked list data structure.

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

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

Data Structures

- class **ConstLinkedListIterator**
- class **ConstReverseIterator**
- class **LinkedListIterator**
- class **ListNode**
- class **ReverseIterator**

Public Member Functions

- **LinkedList** ()
- **LinkedList** (**const LinkedList**< E > &list)
- **LinkedList** (**const Collection**< E > &collection)
- virtual ~**LinkedList** ()
- **LinkedList**< E > & **operator=** (**const LinkedList**< E > &list)
- **LinkedList**< E > & **operator=** (**const Collection**< E > &collection)
- virtual E **get** (int index) **const**

  Gets the element contained at position passed.

  Parameters

index	*The position to get.*

  Returns

  value at index specified.

  Exceptions

IndexOutOfBoundsException	*if the index given is less than zero or greater than the **List** (p. 1286) size.*

  *This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the element using **ListIterator.next** (p. 1209) and returns it.*

- virtual E **set** (int index, **const** E &**element**)

  Replaces the element at the specified position in this list with the specified element.

  Parameters

index	*The index of the element to replace.*
element	*The element to be stored at the specified position.*

  Returns

  the element previously at the specified position.

  Exceptions

IndexOutOfBoundsException	*if the index given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

  *This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the current element using **ListIterator.next** (p. 1209) and replaces it with **ListIterator.set** (p. 1297).*

- virtual bool **add** (**const** E &value)

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

- virtual void **add** (int index, **const** E &value)

  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	*The index at which the specified element is to be inserted.*
element	*The element to be inserted in this **List** (p. 1286).*

  Exceptions

IndexOutOfBoundsException	*if the index is greater than size of the **List** (p. 1286).*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

  *This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it inserts the specified element with **ListIterator.add** (p. 1296).*

- virtual bool **addAll** (**const Collection**< E > &collection)

  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	*The **Collection** (p. 660) whose elements are added to this one.*

  Returns

  true if this collection changed as a result of the call

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
IllegalArgumentException	*if some property of an element prevents it from being added to this collection*
IllegalStateException	*if an element cannot be added at this time due to insertion restrictions.*

  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).

- virtual bool **addAll** (int index, **const Collection**< E > &collection)

  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. 660) 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 given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with listIterator(index)). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using ***ListIterator.add*** *(p. 1296) (to skip over the added element).*

- virtual void **copy** (**const Collection**< E > &collection)

  Renders this ***Collection*** *(p. 660) as a Copy of the given* ***Collection*** *(p. 660).*

- virtual bool **remove** (**const** E &value)

  Removes a single instance of the specified element from the collection.
  More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this* ***Collection*** *(p. 660).*

  Returns

  true if the collection was changed, false otherwise.

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the* ***Collection*** *(p. 660) is a container of pointers and does not allow NULL values.*

  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.

- virtual bool **isEmpty** () **const**

  Returns true if this collection contains no elements.

- virtual int **size** () **const**

  Returns the number of elements in this collection.

- virtual void **clear** ()

  Removes all of the elements from this collection (optional operation).

- virtual bool **contains** (**const** E &value) **const**

  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 (value == NULL ? e == NULL : value == e).

  Parameters

value	*The value to check for presence in the collection.*

  Returns

  true if there is at least one of the elements in the collection

  Exceptions

NullPointerException	*if the* ***Collection*** *(p. 660) contains pointers and the* ***Collection*** *(p. 660) does not allow for NULL elements (optional check).*

  This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

- virtual int **indexOf** (**const** E &value) **const**

  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 int **lastIndexOf** (**const** E &value) **const**

  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 std::vector< E > **toArray** () **const**

  Answers an STL vector containing copies of all elements contained in this ***Collection*** *(p. 660).*

- virtual bool **offer** (**const** E &value)

  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 E **remove** ()

  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 **element** () **const**

  Gets but not removes the element in the head of the queue.

- virtual void **addFirst** (**const** E &value)

  *Inserts an element onto the front of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.*

- virtual void **addLast** (**const** E &value)

  *Inserts an element onto the end of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.*

- virtual E & **getFirst** ()

  *Attempts to fetch a reference to the first element in the **Deque** (p. 926).*

- virtual **const** E & **getFirst** () **const**
- virtual E & **getLast** ()

  *Attempts to fetch a reference to the last element in the **Deque** (p. 926).*

- virtual **const** E & **getLast** () **const**
- virtual bool **offerFirst** (**const** E &**element**)

  *This method attempts to insert the given element into the **Deque** (p. 926) at the front end.*

- virtual bool **offerLast** (**const** E &**element**)

  *This method attempts to insert the given element into the **Deque** (p. 926) at the end.*

- virtual E **removeFirst** ()

  *Removes the topmost element from the **Deque** (p. 926) and returns it.*

- virtual E **removeLast** ()

  *Removes the last element from the **Deque** (p. 926) and returns it.*

- virtual bool **pollFirst** (E &result)

  *Removes the first element from the **Deque** (p. 926) assigns it to the element reference passed.*

- virtual bool **pollLast** (E &result)

  *Removes the last element from the **Deque** (p. 926) assigns it to the element reference passed.*

- virtual bool **peekFirst** (E &result) **const**

  *Retrieves the first element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).*

- virtual bool **peekLast** (E &result) **const**

  *Retrieves the last element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).*

- virtual E **pop** ()

  *Treats this **Deque** (p. 926) as a stack and attempts to pop an element off the top.*

- virtual void **push** (**const** E &**element**)

  Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an IllegalStateException if no space is currently available.

- virtual bool **removeFirstOccurrence** (**const** E &value)

  *Removes the first occurrence of the specified element from this **Deque** (p. 926).*

- virtual bool **removeLastOccurrence** (**const** E &value)

  *Removes the last occurrence of the specified element from this **Deque** (p. 926).*

- virtual **ListIterator**< E > * **listIterator** (int index)
- virtual **ListIterator**< E > * **listIterator** (int index) **const**
- virtual **Iterator**< E > * **descendingIterator** ()

  *Provides an **Iterator** (p. 1209) over this **Collection** (p. 660) that traverses the element in reverse order.*

- virtual **Iterator**< E > * **descendingIterator** () **const**

6.311.1 Detailed Description

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

A complete implementation of the **List** (p. 1286) interface using a doubly linked list data structure.

This class also implements the **Deque** (p. 926) interface providing a common interface for additions into the list at the front and end as well as allowing insertions anywhere in between. This class can be used then to implement other data structures such as Stacks, **Queue** (p. 1723)'s or double ended **Queue** (p. 1723)'s.

The operations on this **List** (p. 1286) object that index a particular element involve iterating over the links of the list from beginning to end, starting from whichever end is closer to the location the operation is to be performed on.

Since

1.0

6.311.2 Constructor & Destructor Documentation

6.311.2.1 template<typename E> decaf::util::LinkedList< E >::LinkedList ( ) [inline]

6.311.2.2 template<typename E> decaf::util::LinkedList< E >::LinkedList (const LinkedList< E > & *list*) [inline]

6.311.2.3 template<typename E> decaf::util::LinkedList< E >::LinkedList (const Collection< E > & *collection*) [inline]

6.311.2.4 template<typename E> virtual decaf::util::LinkedList< E >::~LinkedList ( ) [inline, virtual]

6.311.3 Member Function Documentation

6.311.3.1 template<typename E> virtual bool decaf::util::LinkedList< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Reimplemented from **decaf::util::AbstractList**< **E** > (p. 98).

6.311.3.2 template<typename E> virtual void decaf::util::LinkedList< E >::add (int index , const E & *element*) [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	The index at which the specified element is to be inserted.
element	The element to be inserted in this **List** (p. 1286).

Exceptions

IndexOutOfBounds-Exception	if the index is greater than size of the **List** (p. 1286).
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it inserts the specified element with **ListIterator.add** (p. 1296).

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it inserts the specified element with **ListIterator.add** (p. 1296).

Reimplemented from **decaf::util::AbstractSequentialList**< **E** > (p. 114).

6.311.3.3 template<typename E> virtual bool decaf::util::LinkedList< E >::addAll (const Collection< E > & *collection*) [inline, 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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

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).

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).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 87).

6.311.3.4 template<typename E> virtual bool decaf::util::LinkedList< E >::addAll (int *index*, const Collection< E > & *source*) [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. 660) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with listIterator(index)). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using **ListIterator.add** (p. 1296) (to skip over the added element).

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with listIterator(index)). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using **ListIterator.add** (p. 1296) (to skip over the added element).

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 114).

6.311.3.5 template<typename E> virtual void decaf::util::LinkedList< E >::addFirst (const E & *element*) [inline, virtual]

Inserts an element onto the front of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 926) it is preferable to call offerFirst instead.

Parameters

element	The element to be placed at the front of the **Deque** (p. 926).

Exceptions

IllegalStateException	if the element cannot be added at this time due to capacity restrictions
NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque< E >** (p. 928).

6.311.3.6 template<typename E> virtual void decaf::util::LinkedList< E >::addLast (const E & *element*) [inline, virtual]

Inserts an element onto the end of the **Deque** (p. 926) if possible without violating the implementations capacity restrictions.

For a capacity restricted **Deque** (p. 926) it is preferable to call offerLast instead.

Parameters

element	The element to be placed at the end of the **Deque** (p. 926).

Exceptions

IllegalStateException	if the element cannot be added at this time due to capacity restrictions
NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque< E >** (p. 928).

Referenced by decaf::util::LinkedList< cms::Connection * >::offer().

6.311.3.7 template<typename E> virtual void decaf::util::LinkedList< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractList< E >** (p. 100).

Referenced by decaf::util::LinkedList< cms::Connection * >::copy(), and decaf::util::LinkedList< cms::Connection * >::operator=().

6.311.3.8 template<typename E> virtual bool decaf::util::LinkedList< E >::contains (const E & *value*) const [inline, 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 (value == NULL ? e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 88).

6.311.3.9 template<typename E> virtual void decaf::util::LinkedList< E >::copy (const Collection< E > & *collection*) [inline, virtual]

Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 660)'s clear method.

Parameters

collection	The collection to mirror.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
IllegalStateException	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 89).

6.311.3.10 template<typename E> virtual Iterator<E>* decaf::util::LinkedList< E >::descendingIterator ( ) [inline, virtual]

Provides an **Iterator** (p. 1209) over this **Collection** (p. 660) that traverses the element in reverse order.

Returns

a new **Iterator** (p. 1209) instance that moves from last to first.

Implements **decaf::util::Deque< E >** (p. 929).

Referenced by decaf::util::LinkedList< cms::Connection * >::removeLastOccurrence().

6.311.3.11 template<typename E> virtual Iterator<E>* decaf::util::LinkedList< E >::descendingIterator ( ) const [inline, virtual]

Implements **decaf::util::Deque< E >** (p. 929).

6.311.3.12 template<typename E> virtual E decaf::util::LinkedList< E >::element ( ) const [inline, virtual]

Gets but not removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException *(p. 1537)*	if there is no element in the queue.

Implements **decaf::util::Queue< E >** (p. 1724).

Referenced by decaf::util::LinkedList< cms::Connection * >::set().

6.311.3.13 template<typename E> virtual E decaf::util::LinkedList< E >::get (int *index*) const [inline, virtual]

Gets the element contained at position passed.

Parameters

index	The position to get.

Returns

value at index specified.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the element using **ListIterator.next** (p. 1209) and returns it.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the element using **ListIterator.next** (p. 1209) and returns it.

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 115).

6.311.3.14 template<typename E> virtual E& decaf::util::LinkedList< E >::getFirst ( ) [inline, virtual]

Attempts to fetch a reference to the first element in the **Deque** (p. 926).

This method does not remove the element from the **Deque** (p. 926) but simply returns a reference to it.

Returns

reference to the first element in the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implements **decaf::util::Deque< E >** (p. 929).

6.311.3.15 template<typename E> virtual const E& decaf::util::LinkedList< E >::getFirst ( ) const [inline, virtual]

Implements **decaf::util::Deque< E >** (p. 930).

6.311.3.16 template<typename E> virtual E& decaf::util::LinkedList< E >::getLast ( ) [inline, virtual]

Attempts to fetch a reference to the last element in the **Deque** (p. 926).

This method does not remove the element from the **Deque** (p. 926) but simply returns a reference to it.

Returns

reference to the last element in the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implements **decaf::util::Deque< E >** (p. 930).

6.311.3.17 template<typename E> virtual const E& decaf::util::LinkedList< E >::getLast ( ) const [inline, virtual]

Implements **decaf::util::Deque< E >** (p. 931).

6.311.3.18 template<typename E> virtual int decaf::util::LinkedList< E >::indexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Reimplemented from **decaf::util::AbstractList< E >** (p. 100).

Referenced by decaf::util::LinkedList< cms::Connection * >::contains().

6.311.3.19 template<typename E> virtual bool decaf::util::LinkedList< E >::isEmpty ( ) const [inline, virtual]

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 1286) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

6.311.3.20 template<typename E> virtual int decaf::util::LinkedList< E >::lastIndexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Reimplemented from **decaf::util::AbstractList< E >** (p. 102).

6.311.3.21 template<typename E> virtual ListIterator<E>* decaf::util::LinkedList< E >::listIterator (int *index*) [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

IndexOutOfBounds-Exception	if the index is out of range (index < 0 || index > **size()** (p. 1286))

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 116).

6.311.3.22 template<typename E> virtual ListIterator<E>* decaf::util::LinkedList< E >::listIterator (int index) const [inline, virtual]

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 117).

6.311.3.23 template<typename E> virtual bool decaf::util::LinkedList< E >::offer (const E & *value*) [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. 1723) implementation does not allow Null values to be inserted into the **Queue** (p. 1723).
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 1724).

6.311.3.24 template<typename E> virtual bool decaf::util::LinkedList< E >::offerFirst (const E & *element*) [inline, virtual]

This method attempts to insert the given element into the **Deque** (p. 926) at the front end.

Unlike the addFirst method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

element	The element to add to this **Deque** (p. 926).

Returns

true if the element was added, false otherwise.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque< E >** (p. 931).

6.311.3.25 template<typename E> virtual bool decaf::util::LinkedList< E >::offerLast (const E & *element*) [inline, virtual]

This method attempts to insert the given element into the **Deque** (p. 926) at the end.

Unlike the addLast method that throws an exception if it cannot insert the element due to capacity restrictions etc this method returns false if it cannot insert the element.

Parameters

element	The element to add to this **Deque** (p. 926).

Returns

true if the element was added, false otherwise.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque< E >** (p. 931).

6.311.3.26 template<typename E> LinkedList<E>& decaf::util::LinkedList< E >::operator= (const LinkedList< E > & *list*) [inline]

6.311.3.27 template<typename E> LinkedList<E>& decaf::util::LinkedList< E >::operator= (const Collection< E > & *collection*) [inline]

6.311.3.28 template<typename E> virtual bool decaf::util::LinkedList< 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. 1725).

6.311.3.29 template<typename E> virtual bool decaf::util::LinkedList< E >::peekFirst (E & *value*) const [inline, virtual]

Retrieves the first element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).

If this call is successful it returns true. Unlike getFirst this method does not throw an exception if the **Deque** (p. 926) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implements **decaf::util::Deque< E >** (p. 932).

6.311.3.30 template<typename E> virtual bool decaf::util::LinkedList< E >::peekLast (E & *value*) const [inline, virtual]

Retrieves the last element contained in this **Deque** (p. 926) and assigns its value to the reference value passed the value however is not removed from the **Deque** (p. 926).

If this call is successful it returns true. Unlike getLast this method does not throw an exception if the **Deque** (p. 926) is empty.

Returns

true if an element was assigned to the reference passed, false otherwise.

Implements **decaf::util::Deque< E >** (p. 932).

6.311.3.31 template<typename E> virtual bool decaf::util::LinkedList< 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. 1723) 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. 1725).

6.311.3.32 template<typename E> virtual bool decaf::util::LinkedList< E >::pollFirst (E & *element*) [inline, virtual]

Removes the first element from the **Deque** (p. 926) assigns it to the element reference passed.

Parameters

element	Reference to an variable that can be assigned the value of the head of this **Deque** (p. 926).

Returns

true if an element was available to remove, false otherwise.

Implements **decaf::util::Deque**< **E** > (p. 933).

6.311.3.33 template<typename E> virtual bool decaf::util::LinkedList< E >::pollLast (E & *element*) [inline, virtual]

Removes the last element from the **Deque** (p. 926) assigns it to the element reference passed.

Parameters

element	Reference to an variable that can be assigned the value of the tail of this **Deque** (p. 926).

Returns

true if an element was available to remove, false otherwise.

Implements **decaf::util::Deque**< **E** > (p. 933).

6.311.3.34 template<typename E> virtual E decaf::util::LinkedList< E >::pop ( ) [inline, virtual]

Treats this **Deque** (p. 926) as a stack and attempts to pop an element off the top.

If there's no element to pop then a NuSuchElementException is thrown, otherwise the top element is removed and assigned to the reference passed.

This operation performs the same basic function as the removeFirst method.

Returns

the element at the front of this deque which would be the top of a stack.

Exceptions

NoSuchElementException *(p. 1537)*	if there is nothing on the top of the stack.

Implements **decaf::util::Deque**< **E** > (p. 933).

6.311.3.35 template<typename E> virtual void decaf::util::LinkedList< E >::push (const E & *element*) [inline, virtual]

Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, otherwise it throwing an IllegalStateException if no space is currently available.

This method performs the same basic operation as the addFirst method.

Parameters

element	The element to be pushed onto the **Deque** (p. 926).

Exceptions

IllegalStateException	if the element cannot be added at this time due to capacity restrictions
NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.
IllegalArgumentException	if some property of the specified element prevents it from being added to this deque.

Implements **decaf::util::Deque< E >** (p. 934).

6.311.3.36 template<typename E> virtual bool decaf::util::LinkedList< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 92).

6.311.3.37 template<typename E> virtual E decaf::util::LinkedList< E >::remove ( ) [inline, virtual]

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException *(p. 1537)*	if there is no element in the queue.

Implements **decaf::util::Queue< E >** (p. 1725).

6.311.3.38 template<typename E> virtual E decaf::util::LinkedList< E >::removeFirst ( ) [inline, virtual]

Removes the topmost element from the **Deque** (p. 926) and returns it.

Unlike the pollFirst method this method throws a NuSuchElementException if the **Deque** (p. 926) is empty.

Returns

the element at the Head of the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implements **decaf::util::Deque< E >** (p. 934).

6.311.3.39 template<typename E> virtual bool decaf::util::LinkedList< E >::removeFirstOccurrence (const E & *value*) [inline, virtual]

Removes the first occurrence of the specified element from this **Deque** (p. 926).

If there is no matching element then the **Deque** (p. 926) is left unchanged.

Parameters

value	The value to be removed from this **Deque** (p. 926).

Returns

true if the **Deque** (p. 926) was modified as a result of this operation.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.

Implements **decaf::util::Deque< E >** (p. 935).

Referenced by decaf::util::LinkedList< cms::Connection * >::remove().

6.311.3.40 template<typename E> virtual E decaf::util::LinkedList< E >::removeLast ( ) [inline, virtual]

Removes the last element from the **Deque** (p. 926) and returns it.

Unlike the pollLast method this method throws a NuSuchElementException if the **Deque** (p. 926) is empty.

Returns

the element at the Tail of the **Deque** (p. 926).

Exceptions

NoSuchElementException *(p. 1537)*	if the **Deque** (p. 926) is empty.

Implements **decaf::util::Deque< E >** (p. 935).

6.311.3.41 template<typename E> virtual bool decaf::util::LinkedList< E >::removeLastOccurrence (const E & *value*) [inline, virtual]

Removes the last occurrence of the specified element from this **Deque** (p. 926).

If there is no matching element then the **Deque** (p. 926) is left unchanged.

Parameters

value	The value to be removed from this **Deque** (p. 926).

Returns

true if the **Deque** (p. 926) was modified as a result of this operation.

Exceptions

NullPointerException	if the specified element is NULL and this deque is a **Collection** (p. 660) of pointers and does not permit null elements.

Implements **decaf::util::Deque< E >** (p. 936).

6.311.3.42 template<typename E> virtual E decaf::util::LinkedList< E >::set (int index , const E & *element*) [inline, virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters

index	The index of the element to replace.
element	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the current element using **ListIterator.next** (p. 1209) and replaces it with **ListIterator.set** (p. 1297).

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the current element using **ListIterator.next** (p. 1209) and replaces it with **ListIterator.set** (p. 1297).

Reimplemented from **decaf::util::AbstractSequentialList< E >** (p. 117).

6.311.3.43 template<typename E> virtual int decaf::util::LinkedList< 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. 669).

6.311.3.44 template<typename E> virtual std::vector<E> decaf::util::LinkedList< E >::toArray ( ) const [inline, virtual]

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).

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. 660)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 94).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**LinkedList.h**

6.312 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

- **List** ()
- virtual ~**List** ()
- virtual **ListIterator**< E > * **listIterator** ()=0
- virtual **ListIterator**< E > * **listIterator** () **const** =0
- virtual **ListIterator**< E > * **listIterator** (int index)=0
- virtual **ListIterator**< E > * **listIterator** (int index) **const** =0
- virtual int **indexOf** (**const** E &value) **const** =0

  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 int **lastIndexOf** (**const** E &value) **const** =0

  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** (int index) **const** =0

  Gets the element contained at position passed.
- virtual E **set** (int index, **const** E &element)=0

  Replaces the element at the specified position in this list with the specified element.
- virtual void **add** (int index, **const** E &element)=0

  Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (int index, **const Collection**< E > &source)=0

  Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual E **removeAt** (int index)=0

  Removes the element at the specified position in this list.

6.312.1 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.

6.312.2 Constructor & Destructor Documentation

6.312.2.1 template<typename E> decaf::util::List< E >::List ( ) [inline]

6.312.2.2 template<typename E> virtual decaf::util::List< E >::~List ( ) [inline, virtual]

6.312.3 Member Function Documentation

6.312.3.1 template<typename E> virtual void decaf::util::List< E >::add (int *index,* const E & *element*) [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	The index at which the specified element is to be inserted.
element	The element to be inserted in this **List** (p. 1286).

Exceptions

IndexOutOfBounds-Exception	if the index is greater than size of the **List** (p. 1286).
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 801), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 801), **decaf::util::StlList< E >** (p. 1970), **decaf::util::ArrayList< E >** (p. 348), **decaf::util::LinkedList< E >** (p. 1272), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1272), **decaf::util::LinkedList< CompositeTask * >** (p. 1272), **decaf::util::LinkedList< URI >** (p. 1272), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1272), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1272), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1272), **decaf::util::LinkedList< Pointer< Command > >** (p. 1272), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1272), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1272), **decaf::util::LinkedList< cms::Destination * >** (p. 1272), **decaf::util::LinkedList< cms::Session * >** (p. 1272), **decaf::util::LinkedList< cms::Connection * >** (p. 1272), **decaf::util::AbstractSequentialList< E >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 114), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 114), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 114), **decaf::util::AbstractSequentialList< URI >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 114), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 114), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 114), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 114), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 114), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 114).

6.312.3.2 template<typename E> virtual bool decaf::util::List< E >::addAll (int *index*, const Collection< E > & *source*) [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. 660) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 802), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 802), **decaf::util::StlList< E >** (p. 1971), **decaf::util::AbstractList< E >** (p. 99), **decaf::util::AbstractList< Pointer< Transport > >** (p. 99), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 99), **decaf::util::AbstractList< CompositeTask * >** (p. 99), **decaf::util::AbstractList< URI >** (p. 99), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 99), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 99), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 99), **decaf::util::AbstractList< Pointer< Command > >** (p. 99), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 99), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 99), **decaf::util::AbstractList< cms::Destination * >** (p. 99), **decaf::util::AbstractList< cms::Session * >** (p. 99), **decaf::util::AbstractList< cms::Connection * >** (p. 99), **decaf::util::ArrayList< E >** (p. 349), **decaf::util::LinkedList< E >** (p. 1273), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1273), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1273), **decaf::util::LinkedList< CompositeTask * >** (p. 1273), **decaf::util::LinkedList< URI >** (p. 1273), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1273), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1273), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1273), **decaf::util::LinkedList< Pointer< Command > >** (p. 1273), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1273), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1273), **decaf::util::LinkedList< cms::Destination * >** (p. 1273), **decaf::util::LinkedList< cms::Session * >** (p. 1273), **decaf::util::LinkedList< cms::Connection * >** (p. 1273), **decaf::util::AbstractSequentialList< E >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 114), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 114), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 114), **decaf::util::AbstractSequentialList< URI >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 114), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 114), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 114), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 114), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 114), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 114), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 114).

6.312.3.3 template<typename E> virtual E decaf::util::List< E >::get (int *index*) const [pure virtual]

Gets the element contained at position passed.

Parameters

index	The position to get.

Returns

value at index specified.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 805), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 805), **decaf::util::StlList< E >** (p. 1973), **decaf::util::ArrayList< E >** (p. 351), **decaf::util::LinkedList< E >** (p. 1276), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1276), **decaf::util::LinkedList< CompositeTask * >** (p. 1276), **decaf::util::LinkedList< URI >** (p. 1276), **decaf::util::LinkedList< Pointer< MessageDispatch**

> > (p. 1276), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1276), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1276), **decaf::util::LinkedList< Pointer< Command > >** (p. 1276), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1276), **decaf::util::LinkedList< cms::Destination * >** (p. 1276), **decaf::util::LinkedList< cms::Session * >** (p. 1276), **decaf::util::LinkedList< cms::Connection * >** (p. 1276), **decaf::util::AbstractSequentialList< E >** (p. 115), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 115), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 115), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 115), **decaf::util::AbstractSequentialList< URI >** (p. 115), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 115), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 115), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 115), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 115), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 115), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 115), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 115), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 115), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 115).

6.312.3.4 template<typename E> virtual int decaf::util::List< E >::indexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implemented in **decaf::util::StlList< E >** (p. 1974), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 805), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 805), **decaf::util::AbstractList< E >** (p. 100), **decaf::util::AbstractList< Pointer< Transport > >** (p. 100), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 100), **decaf::util::AbstractList< CompositeTask * >** (p. 100), **decaf::util::AbstractList< URI >** (p. 100), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 100), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 100), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 100), **decaf::util::AbstractList< Pointer< Command > >** (p. 100), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 100), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 100), **decaf::util::AbstractList< cms::Destination * >** (p. 100), **decaf::util::AbstractList< cms::Session * >** (p. 100), **decaf::util::AbstractList< cms::Connection * >** (p. 100), **decaf::util::ArrayList< E >** (p. 352), **decaf::util::LinkedList< E >** (p. 1277), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1277), **decaf::util::LinkedList< CompositeTask * >** (p. 1277), **decaf::util::LinkedList< URI >** (p. 1277), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1277), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1277), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1277), **decaf::util::LinkedList< Pointer< Command > >** (p. 1277), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1277), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1277), **decaf::util::LinkedList< cms::Destination * >** (p. 1277), **decaf::util::LinkedList< cms::Session * >** (p. 1277), and **decaf::util::LinkedList< cms::Connection * >** (p. 1277).

6.312.3.5 template<typename E> virtual int decaf::util::List< E >::lastIndexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Implemented in **decaf::util::StlList< E >** (p. 1975), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 806), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 806), **decaf::util::AbstractList< E >** (p. 102), **decaf::util::AbstractList< Pointer< Transport > >** (p. 102), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 102), **decaf::util::AbstractList< CompositeTask * >** (p. 102), **decaf::util::AbstractList< URI >** (p. 102), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 102), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 102), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 102), **decaf::util::AbstractList< Pointer< Command > >** (p. 102), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 102), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 102), **decaf::util::AbstractList< cms::Destination * >** (p. 102), **decaf::util::AbstractList< cms::Session * >** (p. 102), **decaf::util::AbstractList< cms::Connection * >** (p. 102), **decaf::util::ArrayList< E >** (p. 352), **decaf::util::LinkedList< E >** (p. 1278), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1278), **decaf::util::LinkedList< CompositeTask * >** (p. 1278), **decaf::util::LinkedList< URI >** (p. 1278), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1278), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1278), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1278), **decaf::util::LinkedList< Pointer< Command > >** (p. 1278), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1278), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1278), **decaf::util::LinkedList< cms::Destination * >** (p. 1278), **decaf::util::LinkedList< cms::Session * >** (p. 1278), and **decaf::util::LinkedList< cms::Connection * >** (p. 1278).

6.312.3.6 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::concurrent::CopyOnWriteArrayList< E >** (p. 807), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 807), **decaf::util::AbstractList< E >** (p. 102), **decaf::util::AbstractList< Pointer< Transport > >** (p. 102), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 102), **decaf::util::AbstractList< CompositeTask * >** (p. 102), **decaf::util::AbstractList< URI >** (p. 102), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 102), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 102), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 102), **decaf::util::AbstractList< Pointer< Command > >** (p. 102), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 102), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 102), **decaf::util::AbstractList< cms::Destination * >** (p. 102), **decaf::util::AbstractList< cms::Session * >** (p. 102), **decaf::util::AbstractList< cms::Connection * >** (p. 102), **decaf::util::StlList< E >** (p. 1975), **decaf::util::AbstractSequentialList< E >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 116), **decaf::util::AbstractSequentialList<**

CompositeTask ∗ > (p. 116), **decaf::util::AbstractSequentialList**< **URI** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 116), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 116), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** ∗ > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Destination** ∗ > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Session** ∗ > (p. 116), and **decaf::util::AbstractSequentialList**< **cms::Connection** ∗ > (p. 116).

6.312.3.7 template<typename E> virtual ListIterator<E>∗ decaf::util::List< E >::listIterator ( ) const [pure virtual]

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 807), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** ∗ > (p. 807), **decaf::util::AbstractList**< **E** > (p. 103), **decaf::util::AbstractList**< **Pointer**< **Transport** > > (p. 103), **decaf::util::AbstractList**< **cms::MessageConsumer** ∗ > (p. 103), **decaf::util::AbstractList**< **CompositeTask** ∗ > (p. 103), **decaf::util::AbstractList**< **URI** > (p. 103), **decaf::util::AbstractList**< **Pointer**< **MessageDispatch** > > (p. 103), **decaf::util::AbstractList**< **Pointer**< **DestinationInfo** > > (p. 103), **decaf::util::AbstractList**< **PrimitiveValueNode** > (p. 103), **decaf::util::AbstractList**< **Pointer**< **Command** > > (p. 103), **decaf::util::AbstractList**< **Pointer**< **BackupTransport** > > (p. 103), **decaf::util::AbstractList**< **cms::MessageProducer** ∗ > (p. 103), **decaf::util::AbstractList**< **cms::Destination** ∗ > (p. 103), **decaf::util::AbstractList**< **cms::Session** ∗ > (p. 103), **decaf::util::AbstractList**< **cms::Connection** ∗ > (p. 103), **decaf::util::StlList**< **E** > (p. 1975), **decaf::util::AbstractSequentialList**< **E** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **Transport** > > (p. 116), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** ∗ > (p. 116), **decaf::util::AbstractSequentialList**< **CompositeTask** ∗ > (p. 116), **decaf::util::AbstractSequentialList**< **URI** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 116), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 116), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 116), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** ∗ > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Destination** ∗ > (p. 116), **decaf::util::AbstractSequentialList**< **cms::Session** ∗ > (p. 116), and **decaf::util::AbstractSequentialList**< **cms::Connection** ∗ > (p. 116).

6.312.3.8 template<typename E> virtual ListIterator<E>∗ decaf::util::List< E >::listIterator (int *index*) [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

IndexOutOfBounds-Exception	if the index is out of range (index < 0 || index > **size()** (p. 669))

Implemented in **decaf::util::LinkedList**< **E** > (p. 1278), **decaf::util::LinkedList**< **Pointer**< **Transport** > > (p. 1278), **decaf::util::LinkedList**< **cms::MessageConsumer** ∗ > (p. 1278), **decaf::util::LinkedList**< **CompositeTask** ∗ > (p. 1278), **decaf::util::LinkedList**< **URI** > (p. 1278), **decaf::util::LinkedList**< **Pointer**< **MessageDispatch** > > (p. 1278), **decaf::util::LinkedList**< **Pointer**< **DestinationInfo** > > (p. 1278), **decaf::util::LinkedList**< **PrimitiveValueNode** > (p. 1278), **decaf::util::LinkedList**< **Pointer**< **Command** > > (p. 1278), **decaf::util::LinkedList**< **Pointer**< **BackupTransport** > > (p. 1278), **decaf::util::LinkedList**< **cms::MessageProducer** ∗ > (p. 1278), **decaf::util::LinkedList**< **cms::Destination** ∗ > (p. 1278), **decaf::util::LinkedList**< **cms::Session** ∗ > (p. 1278), **decaf::util::LinkedList**< **cms::Connection** ∗ > (p. 1278), **decaf-**

::util::concurrent::CopyOnWriteArrayList< E > (p. 807), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 807), **decaf::util::AbstractList< E >** (p. 103), **decaf::util::AbstractList< Pointer< Transport > >** (p. 103), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 103), **decaf::util::AbstractList< CompositeTask * >** (p. 103), **decaf::util::AbstractList< URI >** (p. 103), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 103), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 103), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 103), **decaf::util::AbstractList< Pointer< Command > >** (p. 103), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 103), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 103), **decaf::util::AbstractList< cms::Destination * >** (p. 103), **decaf::util::AbstractList< cms::Session * >** (p. 103), **decaf::util::AbstractList< cms::Connection * >** (p. 103), **decaf::util::StlList< E >** (p. 1975), **decaf::util::AbstractSequentialList< E >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 116), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 116), **decaf::util::AbstractSequentialList< URI >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 116), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 116), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 116), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 116), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 116), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 116).

6.312.3.9 template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (int *index*) const [pure virtual]

Implemented in **decaf::util::LinkedList< E >** (p. 1279), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1279), **decaf::util::LinkedList< CompositeTask * >** (p. 1279), **decaf::util::LinkedList< URI >** (p. 1279), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1279), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1279), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1279), **decaf::util::LinkedList< Pointer< Command > >** (p. 1279), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1279), **decaf::util::LinkedList< cms::Destination * >** (p. 1279), **decaf::util::LinkedList< cms::Session * >** (p. 1279), **decaf::util::LinkedList< cms::Connection * >** (p. 1279), **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 808), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 808), **decaf::util::AbstractList< E >** (p. 103), **decaf::util::AbstractList< Pointer< Transport > >** (p. 103), **decaf::util::AbstractList< cms::MessageConsumer * >** (p. 103), **decaf::util::AbstractList< CompositeTask * >** (p. 103), **decaf::util::AbstractList< URI >** (p. 103), **decaf::util::AbstractList< Pointer< MessageDispatch > >** (p. 103), **decaf::util::AbstractList< Pointer< DestinationInfo > >** (p. 103), **decaf::util::AbstractList< PrimitiveValueNode >** (p. 103), **decaf::util::AbstractList< Pointer< Command > >** (p. 103), **decaf::util::AbstractList< Pointer< BackupTransport > >** (p. 103), **decaf::util::AbstractList< cms::MessageProducer * >** (p. 103), **decaf::util::AbstractList< cms::Destination * >** (p. 103), **decaf::util::AbstractList< cms::Session * >** (p. 103), **decaf::util::AbstractList< cms::Connection * >** (p. 103), **decaf::util::StlList< E >** (p. 1976), **decaf::util::AbstractSequentialList< E >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 117), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 117), **decaf::util::AbstractSequentialList< URI >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 117), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 117), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 117).

6.312.3.10 template<typename E> virtual E decaf::util::List< E >::removeAt (int *index*) [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

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList**< **E** > (p. 809), **decaf::util::concurrent::CopyOnWriteArrayList**< **ServiceListener** * > (p. 809), **decaf::util::StlList**< **E** > (p. 1976), **decaf::util::AbstractList**< **E** > (p. 104), **decaf::util::AbstractList**< **Pointer**< **Transport** > > (p. 104), **decaf::util::AbstractList**< **cms::MessageConsumer** * > (p. 104), **decaf::util::AbstractList**< **CompositeTask** * > (p. 104), **decaf::util::AbstractList**< **URI** > (p. 104), **decaf::util::AbstractList**< **Pointer**< **MessageDispatch** > > (p. 104), **decaf::util::AbstractList**< **Pointer**< **DestinationInfo** > > (p. 104), **decaf::util::AbstractList**< **PrimitiveValueNode** > (p. 104), **decaf::util::AbstractList**< **Pointer**< **Command** > > (p. 104), **decaf::util::AbstractList**< **Pointer**< **BackupTransport** > > (p. 104), **decaf::util::AbstractList**< **cms::MessageProducer** * > (p. 104), **decaf::util::AbstractList**< **cms::Destination** * > (p. 104), **decaf::util::AbstractList**< **cms::Session** * > (p. 104), **decaf::util::AbstractList**< **cms::Connection** * > (p. 104), **decaf::util::ArrayList**< **E** > (p. 353), **decaf::util::AbstractSequentialList**< **E** > (p. 117), **decaf::util::AbstractSequentialList**< **Pointer**< **Transport** > > (p. 117), **decaf::util::AbstractSequentialList**< **cms::MessageConsumer** * > (p. 117), **decaf::util::AbstractSequentialList**< **CompositeTask** * > (p. 117), **decaf::util::AbstractSequentialList**< **URI** > (p. 117), **decaf::util::AbstractSequentialList**< **Pointer**< **MessageDispatch** > > (p. 117), **decaf::util::AbstractSequentialList**< **Pointer**< **DestinationInfo** > > (p. 117), **decaf::util::AbstractSequentialList**< **PrimitiveValueNode** > (p. 117), **decaf::util::AbstractSequentialList**< **Pointer**< **Command** > > (p. 117), **decaf::util::AbstractSequentialList**< **Pointer**< **BackupTransport** > > (p. 117), **decaf::util::AbstractSequentialList**< **cms::MessageProducer** * > (p. 117), **decaf::util::AbstractSequentialList**< **cms::Destination** * > (p. 117), **decaf::util::AbstractSequentialList**< **cms::Session** * > (p. 117), and **decaf::util::AbstractSequentialList**< **cms::Connection** * > (p. 117).

6.312.3.11 template<typename E> virtual E decaf::util::List< E >::set (int *index*, const E & *element*) [pure virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters

index	The index of the element to replace.
element	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 810), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 810), **decaf::util::StlList< E >** (p. 1977), **decaf::util::ArrayList< E >** (p. 354), **decaf::util::LinkedList< E >** (p. 1285), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1285), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1285), **decaf::util::LinkedList< CompositeTask * >** (p. 1285), **decaf::util::LinkedList< URI >** (p. 1285), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1285), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1285), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1285), **decaf::util::LinkedList< Pointer< Command > >** (p. 1285), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1285), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1285), **decaf::util::LinkedList< cms::Destination * >** (p. 1285), **decaf::util::LinkedList< cms::Session * >** (p. 1285), **decaf::util::LinkedList< cms::Connection * >** (p. 1285), **decaf::util::AbstractSequentialList< E >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Transport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageConsumer * >** (p. 117), **decaf::util::AbstractSequentialList< CompositeTask * >** (p. 117), **decaf::util::AbstractSequentialList< URI >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< MessageDispatch > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< DestinationInfo > >** (p. 117), **decaf::util::AbstractSequentialList< PrimitiveValueNode >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< Command > >** (p. 117), **decaf::util::AbstractSequentialList< Pointer< BackupTransport > >** (p. 117), **decaf::util::AbstractSequentialList< cms::MessageProducer * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Destination * >** (p. 117), **decaf::util::AbstractSequentialList< cms::Session * >** (p. 117), and **decaf::util::AbstractSequentialList< cms::Connection * >** (p. 117).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**List.h**

6.313 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

  Inserts the specified element into the list (optional operation).
- virtual void **set** (**const** E &e)=0

  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.

6.313.1 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. 1210) 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. 1209) or **previous()** (p. 1296).

6.313.2 Constructor & Destructor Documentation

6.313.2.1 template<typename E > virtual decaf::util::ListIterator< E >::~ListIterator ( ) [inline, virtual]

6.313.3 Member Function Documentation

6.313.3.1 template<typename E > virtual void decaf::util::ListIterator< E >::add (const E & *e*) [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 into the **List** (p. 1286).

Exceptions

UnsupportedOperation-Exception	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.

6.313.3.2 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.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 356).

6.313.3.3 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.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 357).

6.313.3.4 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 *(p. 1537)*	if the iteration has no previous element.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 357).

6.313.3.5 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.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator** (p. 357).

6.313.3.6 template<typename E > virtual void decaf::util::ListIterator< E >::set (const E & *e*) [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. 1210) nor **ListIterator.add** (p. 1296) 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

UnsupportedOperation-Exception	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**

6.314 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **LocalTransactionId** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *__value__) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual bool **isLocalTransactionId** () **const**
- 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**

6.314.1 Member Typedef Documentation

6.314.1.1 typedef decaf::lang::PointerComparator<LocalTransactionId> activemq::commands::LocalTransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 2143).

6.314.2 Constructor & Destructor Documentation

6.314.2.1 activemq::commands::LocalTransactionId::LocalTransactionId ( )

6.314.2.2 activemq::commands::LocalTransactionId::LocalTransactionId (const LocalTransactionId & *other*)

6.314.2.3 virtual activemq::commands::LocalTransactionId::~LocalTransactionId ( ) [virtual]

6.314.3 Member Function Documentation

6.314.3.1 virtual LocalTransactionId* activemq::commands::LocalTransactionId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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::TransactionId** (p. 2144).

6.314.3.2 virtual int activemq::commands::LocalTransactionId::compareTo (const LocalTransactionId & *value*) const [virtual]

6.314.3.3 virtual void activemq::commands::LocalTransactionId::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::TransactionId** (p. 2144).

6.314.3.4 virtual bool activemq::commands::LocalTransactionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 2144).

6.314.3.5 virtual bool activemq::commands::LocalTransactionId::equals (const LocalTransactionId & *value*) const [virtual]

6.314.3.6 virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId ( ) const [virtual]

6.314.3.7 virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId ( ) [virtual]

6.314.3.8 virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::TransactionId** (p. 2145).

6.314.3.9 virtual long long activemq::commands::LocalTransactionId::getValue ( ) const [virtual]

6.314.3.10 virtual bool activemq::commands::LocalTransactionId::isLocalTransactionId ( ) const [inline, virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2145).

6.314.3.11 virtual bool activemq::commands::LocalTransactionId::operator< (const LocalTransactionId & *value*) const [virtual]

6.314.3.12 LocalTransactionId& activemq::commands::LocalTransactionId::operator= (const LocalTransactionId & *other*)

6.314.3.13 virtual bool activemq::commands::LocalTransactionId::operator== (const LocalTransactionId & *value*) const [virtual]

6.314.3.14 virtual void activemq::commands::LocalTransactionId::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.314.3.15 virtual void activemq::commands::LocalTransactionId::setValue (long long *value*) [virtual]

6.314.3.16 virtual std::string activemq::commands::LocalTransactionId::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 2145).

6.314.4 Field Documentation

6.314.4.1 Pointer<ConnectionId> activemq::commands::LocalTransactionId::connectionId [protected]

6.314.4.2 const unsigned char activemq::commands::LocalTransactionId::ID_LOCALTRANSACTIONID = 111 [static]

6.314.4.3 long long activemq::commands::LocalTransactionId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**LocalTransactionId.h**

6.315 activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1301).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/LocalTransactionIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual ~**LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.315.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1301).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.315.2 Constructor & Destructor Documentation

6.315.2.1 activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller ( ) [inline]

6.315.2.2 virtual activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller ( ) [inline, virtual]

6.315.3 Member Function Documentation

6.315.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.315.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.315.3.3 virtual void activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2146).

6.315.3.4 virtual void activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2147).

6.315.3.5 virtual int activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2147).

6.315.3.6 virtual void activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2148).

6.315.3.7 virtual void activemq::wireformat::openwire::marshal::generated::LocalTransactionId-Marshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2148).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**LocalTransactionIdMarshaller.h**

6.316 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 intiallyLocked=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.

6.316.1 Detailed Description

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

Since

1.0

6.316.2 Constructor & Destructor Documentation

6.316.2.1 decaf::util::concurrent::Lock::Lock (Synchronizable * *object*, const bool *intiallyLocked* = true)

Constructor - initializes the object member and locks the object if desired.

Parameters

object	The sync object to control
intiallyLocked	If true, the object will automatically be locked.

6.316.2.2 virtual decaf::util::concurrent::Lock::~Lock ( ) [virtual]

Destructor - Unlocks the object if it is locked.

6.316.3 Member Function Documentation

6.316.3.1 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.

6.316.3.2 void decaf::util::concurrent::Lock::lock ( )

Locks the object.

6.316.3.3 void decaf::util::concurrent::Lock::unlock ( )

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Lock.h**

6.317 decaf::util::concurrent::locks::Lock Class Reference

Lock (p. 1305) 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

  Acquires the lock.
- virtual void **lockInterruptibly** ()=0

  Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** ()=0

  Acquires the lock only if it is free at the time of invocation.
- virtual bool **tryLock** (long long time, **const TimeUnit** &unit)=0

  Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** ()=0

  Releases the lock.
- virtual **Condition** * **newCondition** ()=0

  *Returns a new **Condition** (p. 715) instance that is bound to this **Lock** (p. 1305) instance.*

6.317.1 Detailed Description

Lock (p. 1305) 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. 715) 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. 1741).

While the scoping mechanism for synchronized statements makes it much easier 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. 1305) 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. 1305) 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. 1305) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 1308)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 1307), and an attempt to acquire the lock that can timeout (tryLock(long, TimeUnit)).

Note that **Lock** (p. 1305) 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. 1305) 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

6.317.2 Constructor & Destructor Documentation

6.317.2.1 virtual decaf::util::concurrent::locks::Lock::~Lock ( ) [inline, virtual]

6.317.3 Member Function Documentation

6.317.3.1 virtual void decaf::util::concurrent::locks::Lock::lock ( ) [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. 1305) 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. 1305) implementation.

Exceptions

RuntimeException	if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 1752).

6.317.3.2 virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly ( ) [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. 1305) 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. 1305) 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. 1752).

6.317.3.3 virtual Condition* decaf::util::concurrent::locks::Lock::newCondition ( ) [pure virtual]

Returns a new **Condition** (p. 715) instance that is bound to this **Lock** (p. 1305) instance.

Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()** (p. 717) 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. 715) instance depends on the **Lock** (p. 1305) implementation and must be documented by that implementation.

Returns

A new **Condition** (p. 715) instance for this **Lock** (p. 1305) instance the caller must delete the returned **Condition** (p. 715) object when done with it.

Exceptions

RuntimeException	if an error occurs while creating the **Condition** (p. 715).
UnsupportedOperationException	if this **Lock** (p. 1305) implementation does not support conditions

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 1752).

6.317.3.4 virtual bool decaf::util::concurrent::locks::Lock::tryLock ( ) [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. 1305) 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. 1753).

6.317.3.5 virtual bool decaf::util::concurrent::locks::Lock::tryLock (long long *time,* const TimeUnit & *unit*) [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. 1305) 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. 1305) 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. 1754).

6.317.3.6 virtual void decaf::util::concurrent::locks::Lock::unlock ( ) [pure virtual]

Releases the lock.

Implementation Considerations

A **Lock** (p. 1305) 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. 1305) implementation.

Exceptions

RuntimeException	if an error occurs while acquiring the lock.
IllegalMonitorState-Exception	if the current thread is not the owner of the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 1754).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Lock.h**

6.318 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.

6.318.1 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. 1807) 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 wasInterrupted = true; }

waiters.remove(); if(wasInterrupted) // reassert interrupt status on exit current.interrupt(); }

void unlock() { locked.set(false); **LockSupport.unpark** (p. 1312)(waiters.peek()); } };

Since

1.0

6.318.2 Constructor & Destructor Documentation

6.318.2.1 decaf::util::concurrent::locks::LockSupport::~LockSupport ( )

6.318.3 Member Function Documentation

6.318.3.1 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.

6.318.3.2 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

6.318.3.3 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

6.318.3.4 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**

6.319 decaf::util::logging::Logger Class Reference

A **Logger** (p. 1312) object is used to log messages for a specific system or application component.

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

Public Member Functions

- virtual ~**Logger** ()
- **const** std::string & **getName** () **const**

  *Gets the name of this **Logger** (p. 1312).*
- **Logger** * **getParent** () **const**

  *Gets the parent of this **Logger** (p. 1312) which will be the nearest existing **Logger** (p. 1312) in this Loggers namespace.*
- void **setParent** (**Logger** *parent)

  ***Set** (p. 1857) the parent for this **Logger** (p. 1312).*
- void **addHandler** (**Handler** *handler)

  *Add a log **Handler** (p. 1085) to receive logging messages.*
- void **removeHandler** (**Handler** *handler)

  *Removes the specified **Handler** (p. 1085) from this logger, ownership of the **Handler** (p. 1085) pointer is returned to the caller.*
- **const** std::list< **Handler** * > & **getHandlers** () **const**

  Gets a vector containing all the handlers that this class has been assigned to use.
- void **setFilter** (**Filter** *filter)

  ***Set** (p. 1857) a filter to control output on this **Logger** (p. 1312).*
- **const Filter** * **getFilter** () **const**

  *Gets the **Filter** (p. 1031) object that this class is using.*
- **Level getLevel** () **const**

  *Get the log **Level** (p. 1253) that has been specified for this **Logger** (p. 1312).*
- void **setLevel** (**const Level** &level)

  ***Set** (p. 1857) the log level specifying which message levels will be logged by this logger.*
- bool **getUseParentHandlers** () **const**

  Discover whether or not this logger is sending its output to its parent logger.
- void **setUseParentHandlers** (bool value)

  *Specify whether or not this logger should send its output to it's parent **Logger** (p. 1312).*
- virtual void **entering** (**const** std::string &blockName, **const** std::string &file, **const** int line)

  Logs an Block Enter message.
- virtual void **exiting** (**const** std::string &blockName, **const** std::string &file, **const** int line)

  Logs an Block Exit message.
- virtual void **severe** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a SEVERE **Level** (p. 1253) Log.*
- virtual void **warning** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a WARN **Level** (p. 1253) Log.*
- virtual void **info** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a INFO **Level** (p. 1253) Log.*
- virtual void **debug** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a DEBUG **Level** (p. 1253) Log.*

- virtual void **config** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a CONFIG **Level** (p. 1253) Log.*
- virtual void **fine** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a FINE **Level** (p. 1253) Log.*
- virtual void **finer** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a FINER **Level** (p. 1253) Log.*
- virtual void **finest** (**const** std::string &file, **const** int line, **const** std::string functionName, **const** std::string &message)

  *Log a FINEST **Level** (p. 1253) Log.*
- virtual void **throwing** (**const** std::string &file, **const** int line, **const** std::string functionName, **const decaf::lang::Throwable** &thrown)

  Log throwing an exception.
- virtual bool **isLoggable** (**const Level** &level) **const**

  Check if a message of the given level would actually be logged by this logger.
- virtual void **log** (**LogRecord** &record)

  *Log a **LogRecord** (p. 1333).*
- virtual void **log** (**const Level** &level, **const** std::string &message)

  Log a message, with no arguments.
- virtual void **log** (**const Level** &levels, **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** (**const 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.

Protected Member Functions

- **Logger** (**const** std::string &name)

  *Creates a new instance of the **Logger** (p. 1312) with the given name.*

6.319.1 Detailed Description

A **Logger** (p. 1312) object is used to log messages for a specific system or application component.

Loggers are normally named, using a hierarchical dot-separated namespace. **Logger** (p. 1312) names can be arbitrary strings, but they should normally be based on the namespace or class name of the logged component, such as **decaf.net** (p. 73) or org.apache.decaf. In addition it is possible to create "anonymous" Loggers that are not stored in the **Logger** (p. 1312) namespace.

Logger (p. 1312) objects may be obtained by calls on one of the getLogger factory methods. These will either create a new **Logger** (p. 1312) or return a suitable existing **Logger** (p. 1312).

Logging messages will be forwarded to registered **Handler** (p. 1085) objects, which can forward the messages to a variety of destinations, including consoles, files, OS logs, etc.

Each **Logger** (p. 1312) keeps track of a "parent" **Logger** (p. 1312), which is its nearest existing ancestor in the **Logger** (p. 1312) namespace.

Each **Logger** (p. 1312) has a "Level" associated with it. This reflects a minimum **Level** (p. 1253) that this logger cares about. If a **Logger** (p. 1312)'s level is set to **Level::INHERIT** (p. 1257), then its effective level is inherited from its parent, which may in turn obtain it recursively from its parent, and so on up the tree.

The log level can be configured based on the properties from the logging configuration file, as described in the description of the **LogManager** (p. 1327) class. However it may also be dynamically changed by calls on the **Logger.setLevel** (p. 1321) method. If a logger's level is changed the change may also affect child loggers, since any child logger that has 'inherit' as its level will inherit its effective level from its parent.

On each logging call the **Logger** (p. 1312) initially performs a cheap check of the request level (e.g. SEVERE or FINE) against the effective log level of the logger. If the request level is lower than the log level, the logging call returns immediately.

After passing this initial (cheap) test, the **Logger** (p. 1312) will allocate a **LogRecord** (p. 1333) to describe the logging message. It will then call a **Filter** (p. 1031) (if present) to do a more detailed check on whether the record should be published. If that passes it will then publish the **LogRecord** (p. 1333) to its output Handlers. By default, loggers also publish to their parent's Handlers, recursively up the tree.

Formatting is the responsibility of the output **Handler** (p. 1085), which will typically call a **Formatter** (p. 1074).

Note that formatting need not occur synchronously. It may be delayed until a **LogRecord** (p. 1333) is actually written to an external sink.

All methods on **Logger** (p. 1312) are thread safe.

Since

1.0

6.319.2 Constructor & Destructor Documentation

6.319.2.1 decaf::util::logging::Logger::Logger (const std::string & *name*) [protected]

Creates a new instance of the **Logger** (p. 1312) with the given name.

The logger will be initially configured with a null **Level** (p. 1253) 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 **decaf.net** (p. 73) or org.apache.-decaf. It may be empty for anonymous Loggers.

6.319.2.2 virtual decaf::util::logging::Logger::~Logger ( ) [virtual]

6.319.3 Member Function Documentation

6.319.3.1 void decaf::util::logging::Logger::addHandler (Handler * *handler*)

Add a log **Handler** (p. 1085) to receive logging messages.

By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p. 1312) is configured with a set of Handlers that essentially act as default handlers for all loggers.

The ownership of the given **Handler** (p. 1085) is passed to the **Logger** (p. 1312) and the **Handler** (p. 1085) will be deleted when this **Logger** (p. 1312) is destroyed unless the caller first calls removeHandler with the same pointer value as was originally given.

Parameters

handler	A Logging **Handler** (p. 1085)

Exceptions

NullPointerException	if the **Handler** (p. 1085) given is NULL.

6.319.3.2 virtual void decaf::util::logging::Logger::config (const std::string & *file,* const int *line,* const std::string *functionName,* const std::string & *message*) [virtual]

Log a CONFIG **Level** (p. 1253) Log.

If the logger is currently enabled for the CONFIG message level then the given message is forwarded to all the registered output **Handler** (p. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

6.319.3.3 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** (p. 1253) 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. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

6.319.3.4 virtual void decaf::util::logging::Logger::entering (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 given information is logged at the **Level::FINER** (p. 1256) log level.

Parameters

blockName	The source block name, (usually ClassName::MethodName, or MethodName).
file	The source file name where this method was called from.
line	The source line number where this method was called from.

6.319.3.5 virtual void decaf::util::logging::Logger::exiting (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 enter, a log record with the given information is logged at the **Level::FINER** (p. 1256) log level.

Parameters

blockName	The source block name, (usually ClassName::MethodName, or MethodName).
file	The source file name where this method was called from.
line	The source line number where this method was called from.

6.319.3.6 virtual void decaf::util::logging::Logger::fine (const std::string & *file,* const int *line,* const std::string *functionName,* const std::string & *message*) [virtual]

Log a FINE **Level** (p. 1253) Log.

If the logger is currently enabled for the FINE message level then the given message is forwarded to all the registered output **Handler** (p. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

6.319.3.7 virtual void decaf::util::logging::Logger::finer (const std::string & *file,* const int *line,* const std::string *functionName,* const std::string & *message*) [virtual]

Log a FINER **Level** (p. 1253) Log.

If the logger is currently enabled for the FINER message level then the given message is forwarded to all the registered output **Handler** (p. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

6.319.3.8 virtual void decaf::util::logging::Logger::finest (const std::string & *file,* const int *line,* const std::string *functionName,* const std::string & *message*) [virtual]

Log a FINEST **Level** (p. 1253) Log.

If the logger is currently enabled for the FINEST message level then the given message is forwarded to all the registered output **Handler** (p. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

6.319.3.9 static Logger* decaf::util::logging::Logger::getAnonymousLogger ( ) [static]

Creates an anonymous logger.

The newly created **Logger** (p. 1312) is not registered in the **LogManager** (p. 1327) 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

6.319.3.10 const Filter* decaf::util::logging::Logger::getFilter ( ) const [inline]

Gets the **Filter** (p. 1031) object that this class is using.

Returns

the **Filter** (p. 1031) in use, (can be NULL).

6.319.3.11 const std::list<Handler*>& decaf::util::logging::Logger::getHandlers ( ) const

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

6.319.3.12 Level decaf::util::logging::Logger::getLevel ( ) const [inline]

Get the log **Level** (p. 1253) that has been specified for this **Logger** (p. 1312).

The result may be the INHERIT level, which means that this logger's effective level will be inherited from its parent.

Returns

the level that is currently set

6.319.3.13 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. 1327) and it will configured to also send logging output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 1327) 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. 145)

Returns

a suitable logger.

6.319.3.14 const std::string& decaf::util::logging::Logger::getName ( ) const [inline]

Gets the name of this **Logger** (p. 1312).

Returns

logger name

6.319.3.15 Logger* decaf::util::logging::Logger::getParent ( ) const [inline]

Gets the parent of this **Logger** (p. 1312) which will be the nearest existing **Logger** (p. 1312) in this Loggers namespace.

If this is the Root **Logger** (p. 1312) than this method returns NULL.

Returns

Pointer to this Loggers nearest parent **Logger** (p. 1312).

6.319.3.16 bool decaf::util::logging::Logger::getUseParentHandlers ( ) const [inline]

Discover whether or not this logger is sending its output to its parent logger.

Returns

true if using Parent Handlers

6.319.3.17 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** (p. 1253) 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. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

6.319.3.18 virtual bool decaf::util::logging::Logger::isLoggable (const Level & *level*) const [virtual]

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.

Parameters

level	- a message logging level

Returns

true if the given message level is currently being logged.

6.319.3.19 virtual void decaf::util::logging::Logger::log (LogRecord & *record*) [virtual]

Log a **LogRecord** (p. 1333).

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. 1333) to be published

6.319.3.20 virtual void decaf::util::logging::Logger::log (const 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. 1085) objects

Parameters

level	the **Level** (p. 1253) to log at
message	the message to log

6.319.3.21 virtual void decaf::util::logging::Logger::log (const Level & *levels,* 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. 1085) objects

Parameters

level	the **Level** (p. 1253) to log at
file	the message to log
line	the line in the file
...	variable length argument to format the message string.

6.319.3.22 virtual void decaf::util::logging::Logger::log (const 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. 1333) which is forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 1333) thrown property, rather than the **LogRecord** (p. 1333) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 1333) message

property.

Parameters

level	the **Level** (p. 1253) 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

6.319.3.23 void decaf::util::logging::Logger::removeHandler (Handler * *handler*)

Removes the specified **Handler** (p. 1085) from this logger, ownership of the **Handler** (p. 1085) pointer is returned to the caller.

Returns silently if the given **Handler** (p. 1085) is not found.

Parameters

handler	The **Handler** (p. 1085) to remove

6.319.3.24 void decaf::util::logging::Logger::setFilter (Filter * *filter*)

Set (p. 1857) a filter to control output on this **Logger** (p. 1312).

After passing the initial "level" check, the **Logger** (p. 1312) will call this **Filter** (p. 1031) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

Parameters

filter	The **Filter** (p. 1031) to use, (can be NULL).

6.319.3.25 void decaf::util::logging::Logger::setLevel (const Level & *level*) [inline]

Set (p. 1857) 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** (p. 1257) can be used to turn off logging.

If the new level is the INHERIT **Level** (p. 1253), it means that this node should inherit its level from its nearest ancestor with a specific (non-INHERIT) level value.

Parameters

level	The new **Level** (p. 1253) value to use when logging.

6.319.3.26 void decaf::util::logging::Logger::setParent (Logger * *parent*) [inline]

Set (p. 1857) the parent for this **Logger** (p. 1312).

This method is used by the **LogManager** (p. 1327) to update a **Logger** (p. 1312) when the namespace changes.

It should not be called from application code.

6.319.3.27 void decaf::util::logging::Logger::setUseParentHandlers (bool *value*) [inline]

Specify whether or not this logger should send its output to it's parent **Logger** (p. 1312).

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

6.319.3.28 virtual void decaf::util::logging::Logger::severe (const std::string & *file,* const int *line,* const std::string *functionName,* const std::string & *message*) [virtual]

Log a SEVERE **Level** (p. 1253) Log.

If the logger is currently enabled for the SEVERE message level then the given message is forwarded to all the registered output **Handler** (p. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

6.319.3.29 virtual void decaf::util::logging::Logger::throwing (const std::string & *file,* const int *line,* const std::string *functionName,* const decaf::lang::Throwable & *thrown*) [virtual]

Log throwing an exception.

This is a convenience method to log that a method is terminating by throwing an exception. The logging is done using the FINER level.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 1333) which is forwarded to all registered output handlers. The **LogRecord** (p. 1333)'s message is set to "THROW".

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
thrown	The Throwable that will be thrown, will be cloned.

6.319.3.30 virtual void decaf::util::logging::Logger::warning (const std::string & *file,* const int *line,* const std::string *functionName,* const std::string & *message*) [virtual]

Log a WARN **Level** (p. 1253) 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. 1085) objects.

Parameters

file	The file name where the log was generated.
line	The line number where the log was generated.
functionName	The name of the function that logged this.
message	The message to log at this **Level** (p. 1253).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Logger.h**

6.320 decaf::util::logging::LoggerHierarchy Class Reference

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

Public Member Functions

- **LoggerHierarchy** ()
- virtual ∼**LoggerHierarchy** ()

6.320.1 Constructor & Destructor Documentation

6.320.1.1 decaf::util::logging::LoggerHierarchy::LoggerHierarchy ( )

6.320.1.2 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**

6.321 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** ()

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char ∗buffer, int size, int offset, int length)

6.321.1 Constructor & Destructor Documentation

6.321.1.1 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.

6.321.1.2 virtual activemq::io::LoggingInputStream::~LoggingInputStream ( ) [virtual]

6.321.2 Member Function Documentation

6.321.2.1 virtual int activemq::io::LoggingInputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.321.2.2 virtual int activemq::io::LoggingInputStream::doReadByte ( ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

The documentation for this class was generated from the following file:

- src/main/activemq/io/**LoggingInputStream.h**

6.322 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** ()

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArrayBounded** (**const** unsigned char *buffer, int size, int offset, int length)

6.322.1 Detailed Description

OutputStream filter that just logs the data being written.

6.322.2 Constructor & Destructor Documentation

6.322.2.1 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.

6.322.2.2 virtual activemq::io::LoggingOutputStream::~LoggingOutputStream ( ) [virtual]

6.322.3 Member Function Documentation

6.322.3.1 virtual void activemq::io::LoggingOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

6.322.3.2 virtual void activemq::io::LoggingOutputStream::doWriteByte (unsigned char *c*) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1039).

The documentation for this class was generated from the following file:

- src/main/activemq/io/**LoggingOutputStream.h**

6.323 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)

  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command)

  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.*
UnsupportedOperation-Exception	*if this method is not implemented by this transport.*

- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command, unsigned int timeout)

  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.*
UnsupportedOperation-Exception	*if this method is not implemented by this transport.*

6.323.1 Detailed Description

A transport filter that logs commands as they are sent/received.

6.323.2 Constructor & Destructor Documentation

6.323.2.1 activemq::transport::logging::LoggingTransport::LoggingTransport (const Pointer< Transport > & *next*)

Constructor.

Parameters

next	- the next **Transport** (p. 2161) in the chain

6.323.2.2 virtual activemq::transport::logging::LoggingTransport::~LoggingTransport ( ) [inline, virtual]

6.323.3 Member Function Documentation

6.323.3.1 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. 2173).

6.323.3.2 virtual void activemq::transport::logging::LoggingTransport::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2173).

6.323.3.3 virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Not supported by this class - throws an exception.

Reimplemented from **activemq::transport::TransportFilter** (p. 2174).

6.323.3.4 virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & *command,* unsigned int *timeout*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Not supported by this class - throws an exception.

Reimplemented from **activemq::transport::TransportFilter** (p. 2174).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/logging/**LoggingTransport.h**

6.324 decaf::util::logging::LogManager Class Reference

There is a single global **LogManager** (p. 1327) 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** ()
- bool **addLogger** (**Logger** *logger)

  Add a named logger.
- **Logger** * **getLogger** (**const** std::string &name)

  *Retrieves or creates a new **Logger** (p. 1312) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*
- int **getLoggerNames** (**const** std::vector< std::string > &names)

  *Gets a list of known **Logger** (p. 1312) Names from this Manager, new loggers added while this method is in progress are not garunteed to be in the list.*
- void **setProperties** (**const util::Properties** &properties)

  *Sets the **Properties** (p. 1705) this **LogManager** (p. 1327) should use to configure its loggers.*
- **const util::Properties** & **getProperties** () **const**

  *Gets a reference to the Logging **Properties** (p. 1705) used by this logger.*
- std::string **getProperty** (**const** std::string &name)

  *Gets the value of a named property of this **LogManager** (p. 1327).*
- void **addPropertyChangeListener** (PropertyChangeListener *listener)

  *Adds a change listener for **LogManager** (p. 1327) **Properties** (p. 1705), adding the same instance of a change event listener does nothing.*
- void **removePropertyChangeListener** (PropertyChangeListener *listener)

  *Removes a properties change listener from the **LogManager** (p. 1327), if the listener is not found of the param is NULL this method returns silently.*
- void **readConfiguration** ()

  Reinitialize the logging properties and reread the logging configuration.
- void **readConfiguration** (**decaf::io::InputStream** *stream)

  *Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 1705) format.*
- void **reset** ()

  Reset the logging configuration.

Static Public Member Functions

- static **LogManager** & **getLogManager** ()

  *Get the global **LogManager** (p. 1327) instance.*

Protected Member Functions

- **LogManager** ()

  Constructor, hidden to protect against direct instantiation.
- **LogManager** (**const LogManager** &manager)

  Copy Constructor.
- void **operator=** (**const LogManager** &manager)

  Assignment operator.

Friends

- class **decaf::lang::Runtime**

6.324.1 Detailed Description

There is a single global **LogManager** (p. 1327) object that is used to maintain a set of shared state about Loggers and log services.

This **LogManager** (p. 1327) object:

- Manages a hierarchical namespace of **Logger** (p. 1312) 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. 1327) object can be retrieved using **LogManager::getLogManager()** (p. 1331). The **LogManager** (p. 1327) object is created during class initialization and cannot subsequently be changed.

∗∗∗TODO∗∗∗∗ By default, the **LogManager** (p. 1327) 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. 1327) 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. 1327).

If "decaf.util.logging.config.class" property is not set, then the "decaf.util.logging.config.file" system property can be used to specify a properties file (in **decaf.util.Properties** (p. 1705) 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. 1327) will read its initial configuration from a properties file "lib/logging.properties" in the working 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. 1312) (the **Logger** (p. 1312) named ""). Each class name must be for a **Handler** (p. 1085) 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. 1085) 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. 1085) 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. 1327) object are multi-thread safe.

Since

1.0

6.324.2 Constructor & Destructor Documentation

6.324.2.1 virtual decaf::util::logging::LogManager::~LogManager ( ) [virtual]

6.324.2.2 decaf::util::logging::LogManager::LogManager ( ) [protected]

Constructor, hidden to protect against direct instantiation.

6.324.2.3 decaf::util::logging::LogManager::LogManager (const LogManager & *manager*) [protected]

Copy Constructor.

Parameters

manager	the Manager to copy

6.324.3 Member Function Documentation

6.324.3.1 bool decaf::util::logging::LogManager::addLogger (Logger * *logger*)

Add a named logger.

This does nothing and returns false if a logger with the same name is already registered.

The **Logger** (p. 1312) factory methods call this method to register each newly created **Logger** (p. 1312).

Parameters

logger	The new **Logger** (p. 1312) instance to add to this **LogManager** (p. 1327).

Exceptions

NullPointerException	if logger is NULL.
IllegalArgumentException	if the logger has no name.

6.324.3.2 void decaf::util::logging::LogManager::addPropertyChangeListener (PropertyChangeListener * *listener*)

Adds a change listener for **LogManager** (p. 1327) **Properties** (p. 1705), adding the same instance of a change event listener does nothing.

Parameters

listener	The PropertyChangeListener to add (can be NULL).

6.324.3.3 Logger* decaf::util::logging::LogManager::getLogger (const std::string & *name*)

Retrieves or creates a new **Logger** (p. 1312) 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. 1312).

6.324.3.4 int decaf::util::logging::LogManager::getLoggerNames (const std::vector< std::string > & *names*)

Gets a list of known **Logger** (p. 1312) Names from this Manager, new loggers added while this method is in progress are not garunteed to be in the list.

Parameters

names	STL Vector to hold string logger names.

Returns

names count of how many loggers were inserted.

6.324.3.5 static LogManager& decaf::util::logging::LogManager::getLogManager ( ) [static]

Get the global **LogManager** (p. 1327) instance.

Returns

A reference to the global **LogManager** (p. 1327) instance.

6.324.3.6 const util::Properties& decaf::util::logging::LogManager::getProperties ( ) const [inline]

Gets a reference to the Logging **Properties** (p. 1705) used by this logger.

Returns

The **Logger** (p. 1312) **Properties** (p. 1705) Pointer

6.324.3.7 std::string decaf::util::logging::LogManager::getProperty (const std::string & *name*)

Gets the value of a named property of this **LogManager** (p. 1327).

Parameters

name	The name of the Property to retrieve.

Returns

the value of the property

6.324.3.8 void decaf::util::logging::LogManager::operator= (const LogManager & *manager*) [protected]

Assignment operator.

Parameters

manager	the manager to assign from

6.324.3.9 void decaf::util::logging::LogManager::readConfiguration ( )

Reinitialize the logging properties and reread the logging configuration.

The same rules are used for locating the configuration properties as are used at startup. So normally the logging properties will be re-read from the same file that was used at startup.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 1321), if the target **Logger** (p. 1312) exists.

A PropertyChangeEvent will be fired after the properties are read.

Exceptions

IOException	if an I/O error occurs.

6.324.3.10 void decaf::util::logging::LogManager::readConfiguration (decaf::io::InputStream * *stream*)

Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 1705) format.

A PropertyChangeEvent will be fired after the properties are read.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 1321), if the target **Logger** (p. 1312) exists.

Parameters

stream	The InputStream to read the **Properties** (p. 1705) from.

Exceptions

NullPointerException	if stream is NULL.
IOException	if an I/O error occurs.

6.324.3.11 void decaf::util::logging::LogManager::removePropertyChangeListener (PropertyChangeListener * *listener*)

Removes a properties change listener from the **LogManager** (p. 1327), if the listener is not found of the param is NULL this method returns silently.

Parameters

listener	The PropertyChangeListener to remove from the listeners set.

6.324.3.12 void decaf::util::logging::LogManager::reset ( )

Reset the logging configuration.

For all named loggers, the reset operation removes and closes all Handlers and (except for the root logger) sets the level to INHERIT. The root logger's level is set to **Level::INFO** (p. 1256).

6.324.3.13 void decaf::util::logging::LogManager::setProperties (const util::Properties & *properties*)

Sets the **Properties** (p. 1705) this **LogManager** (p. 1327) should use to configure its loggers.

Once set a properties change event is fired.

Parameters

properties	Pointer to read the configuration from

6.324.4 Friends And Related Function Documentation

6.324.4.1 friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogManager.h**

6.325 decaf::util::logging::LogRecord Class Reference

LogRecord (p. 1333) objects are used to pass logging requests between the logging framework and individual log Handlers.

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

Public Member Functions

- **LogRecord** ()
- virtual ~**LogRecord** ()
- **Level getLevel** () **const**

  *Get **Level** (p. 1253) of this log record.*
- void **setLevel** (**Level** value)

  ***Set** (p. 1857) the **Level** (p. 1253) of this Log Record.*
- **const** std::string & **getLoggerName** () **const**

  *Gets the Source **Logger** (p. 1312)'s Name.*
- void **setLoggerName** (**const** std::string &loggerName)

  *Sets the Source **Logger** (p. 1312)'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 int **getSourceLine** () **const**

Gets the Source Log line number.

- void **setSourceLine** (unsigned int 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.

- long long **getTimestamp** () **const**

  Gets the time in mills that this message was logged.

- void **setTimestamp** (long long timeStamp)

  Sets the time in mills that this message was logged.

- long long **getTreadId** () **const**

  Gets the Thread Id where this Log was created.

- void **setTreadId** (long long threadId)

  Sets the Thread Id where this Log was created.

- **decaf::lang::Throwable** ∗ **getThrown** () **const**

  *Gets any Throwable associated with this **LogRecord** (p. 1333).*

- void **setThrown** (**decaf::lang::Throwable** ∗thrown)

  *Sets the Throwable associated with this **LogRecord** (p. 1333), the pointer becomes the property of this instance of the **LogRecord** (p. 1333) and will be deleted when the record is destroyed.*

6.325.1 Detailed Description

LogRecord (p. 1333) objects are used to pass logging requests between the logging framework and individual log Handlers.

When a **LogRecord** (p. 1333) is passed into the logging framework it logically belongs to the framework and should no longer be used or updated by the client application.

Since

1.0

6.325.2 Constructor & Destructor Documentation

6.325.2.1 decaf::util::logging::LogRecord::LogRecord ( )

6.325.2.2 virtual decaf::util::logging::LogRecord::∼LogRecord ( ) [virtual]

6.325.3 Member Function Documentation

6.325.3.1 Level decaf::util::logging::LogRecord::getLevel ( ) const [inline]

Get **Level** (p. 1253) of this log record.

Returns

Level (p. 1253) enumeration value.

6.325.3.2 const std::string& decaf::util::logging::LogRecord::getLoggerName ( ) const [inline]

Gets the Source **Logger** (p. 1312)'s Name.

Returns

the source loggers name

6.325.3.3 const std::string& decaf::util::logging::LogRecord::getMessage ( ) const [inline]

Gets the Message to be Logged.

Returns

the source logger's message

6.325.3.4 const std::string& decaf::util::logging::LogRecord::getSourceFile ( ) const [inline]

Gets the Source Log File name.

Returns

the source loggers name

6.325.3.5 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

6.325.3.6 unsigned int decaf::util::logging::LogRecord::getSourceLine ( ) const [inline]

Gets the Source Log line number.

Returns

the source loggers line number

6.325.3.7 decaf::lang::Throwable* decaf::util::logging::LogRecord::getThrown ( ) const [inline]

Gets any Throwable associated with this **LogRecord** (p. 1333).

Returns

point to a Throwable instance or Null.

6.325.3.8 long long decaf::util::logging::LogRecord::getTimestamp ( ) const [inline]

Gets the time in mills that this message was logged.

Returns

UTC time in milliseconds

6.325.3.9 long long decaf::util::logging::LogRecord::getTreadId ( ) const [inline]

Gets the Thread Id where this Log was created.

Returns

the source loggers line number

6.325.3.10 void decaf::util::logging::LogRecord::setLevel (Level *value*) [inline]

Set (p. 1857) the **Level** (p. 1253) of this Log Record.

Parameters

value	**Level** (p. 1253) Enumeration Value

6.325.3.11 void decaf::util::logging::LogRecord::setLoggerName (const std::string & *loggerName*) [inline]

Sets the Source **Logger** (p. 1312)'s Name.

Parameters

loggerName	the source loggers name

6.325.3.12 void decaf::util::logging::LogRecord::setMessage (const std::string & *message*) [inline]

Sets the Message to be Logged.

Parameters

message	the source loggers message

6.325.3.13 void decaf::util::logging::LogRecord::setSourceFile (const std::string & *sourceFile*) [inline]

Sets the Source Log File Name.

Parameters

sourceFile	the source loggers name

6.325.3.14 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

6.325.3.15 void decaf::util::logging::LogRecord::setSourceLine (unsigned int *sourceLine*) [inline]

Sets the Source Log line number.

Parameters

sourceLine	the source logger's line number

6.325.3.16 void decaf::util::logging::LogRecord::setThrown (decaf::lang::Throwable * *thrown*) [inline]

Sets the Throwable associated with this **LogRecord** (p. 1333), the pointer becomes the property of this instance of the **LogRecord** (p. 1333) and will be deleted when the record is destroyed.

Parameters

thrown	A pointer to a Throwable that will be associated with this record.

6.325.3.17 void decaf::util::logging::LogRecord::setTimestamp (long long *timeStamp*) [inline]

Sets the time in mills that this message was logged.

Parameters

timeStamp	UTC Time in Milliseconds.

6.325.3.18 void decaf::util::logging::LogRecord::setTreadId (long 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**

6.326 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. 1337) even if there are outstanding references.*

6.326.1 Constructor & Destructor Documentation

6.326.1.1 decaf::util::logging::LogWriter::LogWriter ( )

6.326.1.2 virtual decaf::util::logging::LogWriter::~LogWriter ( ) [virtual]

6.326.2 Member Function Documentation

6.326.2.1 static void decaf::util::logging::LogWriter::destroy ( ) [static]

Forcefully Delete the Instance of this **LogWriter** (p. 1337) even if there are outstanding references.

6.326.2.2 static LogWriter& decaf::util::logging::LogWriter::getInstance ( ) [static]

Get the singleton instance.

6.326.2.3 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	

6.326.2.4 virtual void decaf::util::logging::LogWriter::log (const std::string & *message*) [virtual]

Writes a message to the output destination.

Parameters

message	

6.326.2.5 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**

6.327 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)

  *Constructs a new **Long** (p. 1338) and attempts to convert the given string to an long long value, assigning it to the new object is successful or throwing a NumberFormatException if the string is not a properly formatted long long.*
- virtual ~**Long** ()
- virtual int **compareTo** (**const Long** &l) **const**

  *Compares this **Long** (p. 1338) 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. 1338) 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)

  *Decodes a **String** (p. 2031) into a **Long** (p. 1338).*
- 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)

  Parses the string argument as a signed decimal long.
- static long long **parseLong** (**const** std::string &value, int radix)

  *Returns a **Long** (*p. *1338) 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** (*p. *2031) 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. *1338) instance representing the specified int value.*
- static **Long valueOf** (**const** std::string &value)

  *Returns a **Long** (*p. *1338) object holding the value given by the specified std::string.*
- static **Long valueOf** (**const** std::string &value, int radix)

  *Returns a **Long** (*p. *1338) 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)0x7FFFFFFFFFFFFFFFLL

  The maximum value that the primitive type can hold.
- static **const** long long **MIN_VALUE** = (long long)0x8000000000000000LL

  The minimum value that the primitive type can hold.

6.327.1 Constructor & Destructor Documentation

6.327.1.1 decaf::lang::Long::Long (long long *value*)

Parameters

value	- the primitive long long to wrap

6.327.1.2 decaf::lang::Long::Long (const std::string & *value*)

Constructs a new **Long** (p. 1338) and attempts to convert the given string to an long long value, assigning it to the new object is successful or throwing a NumberFormatException if the string is not a properly formatted long long.

Parameters

value	The string to convert to a primitive type to wrap.

Exceptions

NumberFormatException	if the string is not a a valid 64bit long.

6.327.1.3 virtual decaf::lang::Long::~Long ( ) [inline, virtual]

6.327.2 Member Function Documentation

6.327.2.1 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.

6.327.2.2 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.

Reimplemented from **decaf::lang::Number** (p. 1544).

6.327.2.3 virtual int decaf::lang::Long::compareTo (const Long & *l*) const [virtual]

Compares this **Long** (p. 1338) instance with another.

Parameters

l	- the **Long** (p. 1338) 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** > (p. 687).

6.327.2.4 virtual int decaf::lang::Long::compareTo (const long long & *l*) const [virtual]

Compares this **Long** (p. 1338) instance with another.

Parameters

l	- the **Integer** (p. 1161) 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. 687).

6.327.2.5 static Long decaf::lang::Long::decode (const std::string & *value*) [static]

Decodes a **String** (p. 2031) into a **Long** (p. 1338).

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.parseInteger 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** (p. 2031) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value	- The string to decode

Returns

a **Long** (p. 1338) object containing the decoded value

Exceptions

NumberFomatException	if the string is not formatted correctly.

6.327.2.6 virtual double decaf::lang::Long::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.327.2.7 bool decaf::lang::Long::equals (const Long & *l*) const [inline, virtual]

Parameters

l	- the **Long** (p. 1338) object to compare against.

Returns

true if the two **Integer** (p. 1161) Objects have the same value.

Implements **decaf::lang::Comparable**< **Long** > (p. 688).

6.327.2.8 bool decaf::lang::Long::equals (const long long & *l*) const [inline, virtual]

Parameters

l	- the **Long** (p. 1338) object to compare against.

Returns

true if the two **Integer** (p. 1161) Objects have the same value.

Implements **decaf::lang::Comparable**< **long long** > (p. 688).

6.327.2.9 virtual float decaf::lang::Long::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.327.2.10 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.

6.327.2.11 virtual int decaf::lang::Long::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.327.2.12 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.

Implements **decaf::lang::Number** (p. 1545).

6.327.2.13 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.

6.327.2.14 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.

6.327.2.15 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.

6.327.2.16 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.

Implements **decaf::lang::Comparable< Long >** (p. 688).

6.327.2.17 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. 688).

6.327.2.18 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.

Implements **decaf::lang::Comparable**< **Long** > (p. 688).

6.327.2.19 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. 688).

6.327.2.20 static long long decaf::lang::Long::parseLong (const std::string & *value*) [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** (p. 2031) to parse

Returns

long long value

Exceptions

NumberFormatException	on invalid string value

6.327.2.21 static long long decaf::lang::Long::parseLong (const std::string & *value,* int *radix*) [static]

Returns a **Long** (p. 1338) 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. 1338) object that represents the long long value specified by the string.

Parameters

value	- **String** (p. 2031) to parse
radix	- the base encoding of the string

Returns

long long value

Exceptions

NumberFormatException	on invalid string value

6.327.2.22 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.

6.327.2.23 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.

6.327.2.24 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.

6.327.2.25 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.

6.327.2.26 virtual short decaf::lang::Long::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1545).

6.327.2.27 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.

6.327.2.28 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

6.327.2.29 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

6.327.2.30 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

6.327.2.31 std::string decaf::lang::Long::toString ( ) const

Returns

this **Long** (p. 1338) Object as a **String** (p. 2031) Representation

6.327.2.32 static std::string decaf::lang::Long::toString (long long *value*) [static]

Converts the long to a **String** (p. 2031) representation.

Parameters

value	The long to convert to a std::string.

Returns

string representation

6.327.2.33 static std::string decaf::lang::Long::toString (long long *value,* int *radix*) [static]

6.327.2.34 static Long decaf::lang::Long::valueOf (long long *value*) [inline, static]

Returns a **Long** (p. 1338) instance representing the specified int value.

Parameters

value	- the long long to wrap

Returns

the new **Integer** (p. 1161) object wrapping value.

6.327.2.35 static Long decaf::lang::Long::valueOf (const std::string & *value*) [static]

Returns a **Long** (p. 1338) 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. 1161) object that represents the long long value specified by the string.

Parameters

value	- std::string to parse as base 10

Returns

new **Long** (p. 1338) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a decimal long long.

6.327.2.36 static Long decaf::lang::Long::valueOf (const std::string & *value,* int *radix*) [static]

Returns a **Long** (p. 1338) 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. 1338) 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. 1338) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a valid long long.

6.327.3 Field Documentation

6.327.3.1 const long long decaf::lang::Long::MAX_VALUE = (long long)0x7FFFFFFFFFFFFFFFLL [static]

The maximum value that the primitive type can hold.

6.327.3.2 const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL [static]

The minimum value that the primitive type can hold.

6.327.3.3 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**

6.328 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** (int size, bool readOnly=false)

  *Creates a **IntArrayBuffer** (p. 1144) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **LongArrayBuffer** (long long ∗**array**, int size, int offset, int length, bool readOnly=false)

  *Creates a **LongArrayBuffer** (p. 1351) object that wraps the given array.*
- **LongArrayBuffer** (**const decaf::lang::Pointer**< **ByteArrayAdapter** > &**array**, int offset, int length, bool readOnly=false)

  Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **LongArrayBuffer** (**const LongArrayBuffer** &other)

  *Create a **LongArrayBuffer** (p. 1351) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ∼**LongArrayBuffer** ()

- virtual long long * **array** ()

  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. 451).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual int **arrayOffset** ()

  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 longo the backing array where index zero starts.

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual **LongBuffer** * **asReadOnlyBuffer** () **const**

  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.

- virtual **LongBuffer** & **compact** ()

  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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1359).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **LongBuffer** * **duplicate** ()

  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. 451) which the caller owns.*

- virtual long long **get** ()

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. 474)	*if there no more data to return.*

- virtual long long **get** (int index) **const**

Absolute get method.
Reads the value at the given index.

Parameters

index	*The index in the **Buffer** (p. 451) 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, or index is negative.*

- virtual bool **hasArray** () **const**

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. Subclasses 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.

- virtual bool **isReadOnly** () **const**

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

- virtual **LongBuffer** & **put** (long long value)

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. 472)	*if this buffer's current position is not smaller than its limit*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only*

- virtual **LongBuffer** & **put** (int index, long long value)

Writes the given long longs long longo this buffer at the given index.

Parameters

index	*The position in the **Buffer** (p. 451) 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. 1739)	*if this buffer is read-only*

- virtual **LongBuffer** * **slice** () **const**

  *Creates a new **LongBuffer** (p. 1359) 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. 1359) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **LongArrayBuffer** (p. 1351) as Read-Only.*

6.328.1 Constructor & Destructor Documentation

6.328.1.1 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (int *size*, bool *readOnly* = `false`)

Creates a **IntArrayBuffer** (p. 1144) 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

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArguementException	if the capacity value is negative.

6.328.1.2 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (long long * *array*, int *size*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a **LongArrayBuffer** (p. 1351) 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	The actual array to wrap.
size	The size of the given array.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if offset is greater than array capacity.

6.328.1.3 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array,* int *offset,* int *length,* bool *readOnly* = false)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **LongArrayBuffer** (p. 1351) will be that of the remaining capacity of the passed buffer.

Parameters

array	The ByteArrayAdapter to wrap.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if array is NULL
IndexOutOfBounds-Exception	if offset + length is greater than array size.

6.328.1.4 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (const LongArrayBuffer & *other*)

Create a **LongArrayBuffer** (p. 1351) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.

Parameters

other	The **LongArrayBuffer** (p. 1351) this one is to mirror.

6.328.1.5 virtual decaf::internal::nio::LongArrayBuffer::~LongArrayBuffer ( ) [virtual]

6.328.2 Member Function Documentation

6.328.2.1 virtual long long* decaf::internal::nio::LongArrayBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 1361).

6.328.2.2 virtual int decaf::internal::nio::LongArrayBuffer::arrayOffset ( ) [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 longo the backing array where index zero starts.

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 1362).

6.328.2.3 virtual LongBuffer* decaf::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. 1362).

6.328.2.4 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1359).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::LongBuffer** (p. 1362).

6.328.2.5 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** (p. 451) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1363).

6.328.2.6 virtual long long decaf::internal::nio::LongArrayBuffer::get ( ) [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. 474)*	if there no more data to return.

Implements **decaf::nio::LongBuffer** (p. 1363).

6.328.2.7 virtual long long decaf::internal::nio::LongArrayBuffer::get (int *index*) const [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the long long is to be read.

Returns

the long long that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::LongBuffer** (p. 1363).

6.328.2.8 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. Subclasses 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. 1365).

6.328.2.9 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.

Implements **decaf::nio::Buffer** (p. 454).

6.328.2.10 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (long long *value*) [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. 472)*	if this buffer's current position is not smaller than its limit
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1366).

6.328.2.11 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (int *index*, long long *value*) [virtual]

Writes the given long longs long longo this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The long longs to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1367).

6.328.2.12 virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **LongArrayBuffer** (p. 1351) as Read-Only.

Parameters

value	Boolean value, true if this buffer is to be read-only, false otherwise.

6.328.2.13 virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice ( ) const [virtual]

Creates a new **LongBuffer** (p. 1359) 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. 1359) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1367).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**LongArrayBuffer.h**

6.329 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

Returns the long long array that backs this buffer (optional operation).

- virtual int **arrayOffset** ()=0

  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

  Compacts this buffer.

- virtual **LongBuffer** ∗ **duplicate** ()=0

  Creates a new long long buffer that shares this buffer's content.

- virtual long long **get** ()=0

  Relative get method.

- virtual long long **get** (int index) **const** =0

  Absolute get method.

- **LongBuffer** & **get** (std::vector< long long > buffer)

  Relative bulk get method.

- **LongBuffer** & **get** (long long ∗buffer, int size, int offset, int length)

  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)

  This method transfers the long longs remaining in the given source buffer long longo this buffer.

- **LongBuffer** & **put** (**const** long long ∗buffer, int size, int offset, int length)

  This method transfers long longs long longo this buffer from the given source array.

- **LongBuffer** & **put** (std::vector< long long > &buffer)

  This method transfers the entire content of the given source long longs array long longo this buffer.

- virtual **LongBuffer** & **put** (long long value)=0

  Writes the given long longs long longo this buffer at the current position, and then increments the position.

- virtual **LongBuffer** & **put** (int index, long long value)=0

  Writes the given long longs long longo this buffer at the given index.

- virtual **LongBuffer** ∗ **slice** () **const** =0

  *Creates a new **LongBuffer** (p. 1359) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (**const LongBuffer** &value) **const**

- virtual bool **equals** (**const LongBuffer** &value) **const**

- virtual bool **operator==** (**const LongBuffer** &value) **const**

- virtual bool **operator**< (**const LongBuffer** &value) **const**

Static Public Member Functions

- static **LongBuffer** ∗ **allocate** (int **capacity**)

  Allocates a new Double buffer.

- static **LongBuffer** ∗ **wrap** (long long ∗**array**, int size, int offset, int length)

  *Wraps the passed buffer with a new **LongBuffer** (p. 1359).*

- static **LongBuffer** ∗ **wrap** (std::vector< long long > &buffer)

  *Wraps the passed STL long long Vector in a **LongBuffer** (p. 1359).*

Protected Member Functions

- **LongBuffer** (int **capacity**)

  *Creates a **LongBuffer** (p. 1359) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.*

6.329.1 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.

6.329.2 Constructor & Destructor Documentation

6.329.2.1 decaf::nio::LongBuffer::LongBuffer (int *capacity*) [protected]

Creates a **LongBuffer** (p. 1359) 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	The size and limit of the **Buffer** (p. 451) in long longs.

Exceptions

IllegalArguementException	if capacity is negative.

6.329.2.2 virtual decaf::nio::LongBuffer::~LongBuffer ( ) [inline, virtual]

6.329.3 Member Function Documentation

6.329.3.1 static LongBuffer* decaf::nio::LongBuffer::allocate (int *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. 1359) that was allocated, caller owns.

6.329.3.2 virtual long long* decaf::nio::LongBuffer::array ( ) [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. 451).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1355).

6.329.3.3 virtual int decaf::nio::LongBuffer::arrayOffset ( ) [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 longo the backing array where index zero starts.

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1355).

6.329.3.4 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. 1356).

6.329.3.5 virtual LongBuffer& decaf::nio::LongBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1359).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1356).

6.329.3.6 virtual int decaf::nio::LongBuffer::compareTo (const LongBuffer & *value*) const [virtual]

6.329.3.7 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. 451) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1356).

6.329.3.8 virtual bool decaf::nio::LongBuffer::equals (const LongBuffer & *value*) const [virtual]

6.329.3.9 virtual long long decaf::nio::LongBuffer::get ( ) [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. 474)*	if there no more data to return.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1357).

6.329.3.10 virtual long long decaf::nio::LongBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the long long is to be read.

Returns

the long long that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1357).

6.329.3.11 LongBuffer& decaf::nio::LongBuffer::get (std::vector< long long > *buffer*)

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. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length long longs remaining in this buffer.

6.329.3.12 LongBuffer& decaf::nio::LongBuffer::get (long long * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no bytes are transferred and a **BufferUnderflowException** (p. 474) 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	The pointer to an allocated long long buffer to fill.
size	The size of the passed in buffer.
offset	The position in the buffer to start filling.
length	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length long longs remaining in this buffer
NullPolong	longerException if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.329.3.13 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. Subclasses 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. 1357).

6.329.3.14 virtual bool decaf::nio::LongBuffer::operator< (const LongBuffer & *value*) const [virtual]

6.329.3.15 virtual bool decaf::nio::LongBuffer::operator== (const LongBuffer & *value*) const [virtual]

6.329.3.16 LongBuffer& decaf::nio::LongBuffer::put (LongBuffer & *src*)

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. 456), then no long longs are transferred and a **BufferOverflowException** (p. 472) 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. 472)*	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. 1739)*	if this buffer is read-only

6.329.3.17 LongBuffer& decaf::nio::LongBuffer::put (const long long * *buffer*, int *size*, int *offset*, int *length*)

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. 456), then no long longs are transferred and a **BufferOverflowException** (p. 472) 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.
size	The size of the buffer passed.
offset	The offset within the array of the first char 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. 472)*	if there is insufficient space in this buffer
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only
NullPolong	longerException if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.329.3.18 LongBuffer& decaf::nio::LongBuffer::put (std::vector< long long > & *buffer*)

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. 1359).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

6.329.3.19 virtual LongBuffer& decaf::nio::LongBuffer::put (long long *value*) [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. 472)*	if this buffer's current position is not smaller than its limit
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1358).

6.329.3.20 virtual LongBuffer& decaf::nio::LongBuffer::put (int *index*, long long *value*) [pure virtual]

Writes the given long longs long longo this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data
value	The long longs to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1358).

6.329.3.21 virtual LongBuffer* decaf::nio::LongBuffer::slice ( ) const [pure virtual]

Creates a new **LongBuffer** (p. 1359) 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. 1359) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1359).

6.329.3.22 virtual std::string decaf::nio::LongBuffer::toString ( ) const [virtual]

Returns

a std::string describing this object

6.329.3.23 static LongBuffer* decaf::nio::LongBuffer::wrap (long long * *array,* int *size,* int *offset,* int *length*) [static]

Wraps the passed buffer with a new **LongBuffer** (p. 1359).

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.
size	The size of the passed in array.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new **LongBuffer** (p. 1359) that is backed by buffer, caller owns.

Exceptions

NullPointerException	if the array pointer is NULL.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.329.3.24 static LongBuffer* decaf::nio::LongBuffer::wrap (std::vector< long long > & *buffer*) [static]

Wraps the passed STL long long Vector in a **LongBuffer** (p. 1359).

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. 1359) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**LongBuffer.h**

6.330 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** ()

6.330.1 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.

6.330.2 Constructor & Destructor Documentation

6.330.2.1 activemq::util::LongSequenceGenerator::LongSequenceGenerator ( )

6.330.2.2 virtual activemq::util::LongSequenceGenerator::~LongSequenceGenerator ( ) [inline, virtual]

6.330.3 Member Function Documentation

6.330.3.1 long long activemq::util::LongSequenceGenerator::getLastSequenceId ( )

Returns

the last id that was generated.

6.330.3.2 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**

6.331 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 ()

6.331.1 Constructor & Destructor Documentation

6.331.1.1 decaf::net::MalformedURLException::MalformedURLException ( ) throw () [inline]

Default Constructor.

6.331.1.2 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

6.331.1.3 decaf::net::MalformedURLException::MalformedURLException (const MalformedURLException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.331.1.4 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

6.331.1.5 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.

6.331.1.6 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

6.331.1.7 virtual decaf::net::MalformedURLException::~MalformedURLException ( ) throw () [inline, virtual]

6.331.2 Member Function Documentation

6.331.2.1 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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**MalformedURLException.h**

6.332 decaf::util::Map< K, V, COMPARATOR > Class Template Reference

Map (p. 1371) 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

  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 int **size** () **const** =0
- virtual V & **get** (**const** K &key)=0

  *Gets the value mapped to the specified key in the **Map*** (p. *1371).*
- virtual **const** V & **get** (**const** K &key) **const** =0

  *Gets the value mapped to the specified key in the **Map*** (p. *1371).*
- virtual void **put** (**const** K &key, **const** V &value)=0

  Sets the value for the specified key.
- virtual void **putAll** (**const Map**< K, V, COMPARATOR > &other)=0

  *Stores a copy of the Mappings contained in the other **Map*** (p. *1371) in this one.*
- virtual V **remove** (**const** K &key)=0

  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. *1857) view of the mappings contained in this map.*
- virtual std::vector< V > **values** () **const** =0

6.332.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>>class decaf::util::Map< K, V, COMPARATOR >

Map (p. 1371) 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.

6.332.2 Constructor & Destructor Documentation

6.332.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Map ( ) [inline]

Default constructor - does nothing.

6.332.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::~Map ( ) [inline, virtual]

6.332.3 Member Function Documentation

6.332.3.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::clear ( ) [pure virtual]

Removes all keys and values from this map.

Exceptions

UnsupportedOperation-Exception	if this map is unmodifiable.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 705), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 705), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 705), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 705), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 705), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 705), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 705), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1981), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1981), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1981), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1981), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1981), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1981), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1981), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1981), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1981), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1981), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1981), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1981), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1981).

6.332.3.2 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. 706), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1981), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1981), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1981), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1981), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1981), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1981), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1981), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1981), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1981), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1981), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1981), **decaf-**

::util::StlMap< std::string, CachedProducer * > (p. 1981), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1981).

6.332.3.3 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. 706), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1982), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1982), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1982), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1982), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1982), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1982), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1982), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1982), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1982), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1982), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1982), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1982), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1982).

6.332.3.4 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. 707), and **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1982).

6.332.3.5 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. 1371) to compare to this one.

Returns

true if the **Map** (p. 1371) passed is equal in value to this one.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 707), and **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1982).

6.332.3.6 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::Map< K, V, COMPARATOR >::get (const K & *key*) [pure virtual]

Gets the value mapped to the specified key in the **Map** (p. 1371).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.

Parameters

key	The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException *(p. 1537)*	if the key requests doesn't exist in the **Map** (p. 1371).

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 707), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1983), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1983), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1983), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1983), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1983), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1983), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1983), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1983), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1983), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1983), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1983), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1983), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1983).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals(), decaf::util::StlMap< std::string, cms::Topic * >::putAll(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll().

6.332.3.7 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::get (const K & *key*) const [pure virtual]

Gets the value mapped to the specified key in the **Map** (p. 1371).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.

Parameters

key	The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException *(p. 1537)*	if the key requests doesn't exist in the **Map** (p. 1371).

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 708), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1983), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1983), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1983), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1983), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1983), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1983), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1983), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1983), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1983), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1983), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1983), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1983), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1983).

6.332.3.8 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. 1371) contains any element or not, TRUE or FALSE

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 708), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1984), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1984), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1984), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1984), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1984), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1984), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1984), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1984), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1984), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1984), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1984).

6.332.3.9 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. 1857) 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. 1210), **Set.remove** (p. 667), 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. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 708), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1984), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1984), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1984), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1984), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1984), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1984), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1984), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1984), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1984), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1984), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1984).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals(), decaf::util::StlMap< std::string, cms::Topic * >::putAll(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll().

6.332.3.10 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*) [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. 709), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1985), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1985), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1985), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1985), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1985), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1985), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1985), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1985), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1985), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1985), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1985), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1985), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1985).

6.332.3.11 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*) [pure virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 1371) in this one.

Parameters

other	A **Map** (p. 1371) instance whose elements are to all be inserted in this **Map** (p. 1371).

Exceptions

UnsupportedOperation-Exception	If the implementing class does not support the putAll operation.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 710), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1985), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1985), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1985), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1985), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1985), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1985), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1985), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1985), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1985), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1985), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1985), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1985), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1985).

6.332.3.12 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::Map< K, V, COMPARATOR >::remove (const K & *key*) [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 *(p. 1537)*	if this key is not in the **Map** (p. 1371).
UnsupportedOperation-Exception	if this map is unmodifiable.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 711), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1986), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1986), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1986), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1986), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1986), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1986), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1986), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1986), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1986), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1986), and **decaf::util::StlMap< std::string, cms::Topic**

∗ > (p. 1986).

6.332.3.13 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual int 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. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 713), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1986), **decaf::util::StlMap< cms::Session ∗, SessionResolver ∗ >** (p. 1986), **decaf::util::StlMap< std::string, WireFormatFactory ∗ >** (p. 1986), **decaf::util::StlMap< decaf::lang::Runnable ∗, decaf::util::TimerTask ∗ >** (p. 1986), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1986), **decaf::util::StlMap< std::string, cms::Queue ∗ >** (p. 1986), **decaf::util::StlMap< std::string, CachedConsumer ∗ >** (p. 1986), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer ∗, commands::ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< std::string, TransportFactory ∗ >** (p. 1986), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1986), **decaf::util::StlMap< std::string, CachedProducer ∗ >** (p. 1986), and **decaf::util::StlMap< std::string, cms::Topic ∗ >** (p. 1986).

6.332.3.14 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. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 713), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1987), **decaf::util::StlMap< cms::Session ∗, SessionResolver ∗ >** (p. 1987), **decaf::util::StlMap< std::string, WireFormatFactory ∗ >** (p. 1987), **decaf::util::StlMap< decaf::lang::Runnable ∗, decaf::util::TimerTask ∗ >** (p. 1987), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1987), **decaf::util::StlMap< std::string, cms::Queue ∗ >** (p. 1987), **decaf::util::StlMap< std::string, CachedConsumer ∗ >** (p. 1987), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer ∗, commands::ConsumerId::COMPARATOR >** (p. 1987), **decaf::util::StlMap< std::string, TransportFactory ∗ >** (p. 1987), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1987), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1987), **decaf::util::StlMap< std::string, CachedProducer ∗ >** (p. 1987), and **decaf::util::StlMap< std::string, cms::Topic ∗ >** (p. 1987).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Map.h**

6.333 cms::MapMessage Class Reference

A **MapMessage** (p. 1379) 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** () throw ()
- virtual bool **isEmpty** () **const** =0

  *Returns true if there are no values stored in the **MapMessage** (p. 1379) body.*
- virtual std::vector< std::string > **getMapNames** () **const** =0

  *Returns an Enumeration of all the names in the **MapMessage** (p. 1379) object.*
- virtual bool **itemExists** (**const** std::string &name) **const** =0

  *Indicates whether an item exists in this **MapMessage** (p. 1379) object.*
- virtual bool **getBoolean** (**const** std::string &name) **const** =0

  Returns the Boolean value of the Specified name.
- virtual void **setBoolean** (**const** std::string &name, bool value)=0

  Sets a boolean value with the specified name into the Map.
- virtual unsigned char **getByte** (**const** std::string &name) **const** =0

  Returns the Byte value of the Specified name.
- virtual void **setByte** (**const** std::string &name, unsigned char value)=0

  Sets a Byte value with the specified name into the Map.
- virtual std::vector< unsigned char > **getBytes** (**const** std::string &name) **const** =0

  Returns the Bytes value of the Specified name.
- virtual void **setBytes** (**const** std::string &name, **const** std::vector< unsigned char > &value)=0

  Sets a Bytes value with the specified name into the Map.
- virtual char **getChar** (**const** std::string &name) **const** =0

  Returns the Char value of the Specified name.
- virtual void **setChar** (**const** std::string &name, char value)=0

  Sets a Char value with the specified name into the Map.
- virtual double **getDouble** (**const** std::string &name) **const** =0

  Returns the Double value of the Specified name.
- virtual void **setDouble** (**const** std::string &name, double value)=0

  Sets a Double value with the specified name into the Map.
- virtual float **getFloat** (**const** std::string &name) **const** =0

  Returns the Float value of the Specified name.
- virtual void **setFloat** (**const** std::string &name, float value)=0

  Sets a Float value with the specified name into the Map.
- virtual int **getInt** (**const** std::string &name) **const** =0

  Returns the Int value of the Specified name.
- virtual void **setInt** (**const** std::string &name, int value)=0

  Sets a Int value with the specified name into the Map.
- virtual long long **getLong** (**const** std::string &name) **const** =0

  Returns the Long value of the Specified name.
- virtual void **setLong** (**const** std::string &name, long long value)=0

  Sets a Long value with the specified name into the Map.

- virtual short **getShort** (**const** std::string &name) **const** =0

  Returns the Short value of the Specified name.
- virtual void **setShort** (**const** std::string &name, short value)=0

  Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (**const** std::string &name) **const** =0

  Returns the String value of the Specified name.
- virtual void **setString** (**const** std::string &name, **const** std::string &value)=0

  Sets a String value with the specified name into the Map.

6.333.1 Detailed Description

A **MapMessage** (p. 1379) 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. 1379) inherits from the **Message** (p. 1426) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p. 1379), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p. 1490) is thrown. To place the **MapMessage** (p. 1379) back into a state where it can be read from and written to, call the `clearBody` method.

MapMessage (p. 1379) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 640). The String-to-primitive conversions may throw a **MessageFormatException** (p. 1478) 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

1.0

6.333.2 Constructor & Destructor Documentation

6.333.2.1 virtual cms::MapMessage::~MapMessage ( ) throw () [virtual]

6.333.3 Member Function Documentation

6.333.3.1 virtual bool cms::MapMessage::getBoolean (const std::string & *name*) const [pure virtual]

Returns the Boolean value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 211).

6.333.3.2 virtual unsigned char cms::MapMessage::getByte (const std::string & *name*) const [pure virtual]

Returns the Byte value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 212).

6.333.3.3 virtual std::vector<unsigned char> cms::MapMessage::getBytes (const std::string & *name*) const [pure virtual]

Returns the Bytes value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 212).

6.333.3.4 virtual char cms::MapMessage::getChar (const std::string & *name*) const [pure virtual]

Returns the Char value of the Specified name.

Parameters

name	name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 212).

6.333.3.5 virtual double cms::MapMessage::getDouble (const std::string & *name*) const [pure virtual]

Returns the Double value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 213).

6.333.3.6 virtual float cms::MapMessage::getFloat (const std::string & *name*) const [pure virtual]

Returns the Float value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 213).

6.333.3.7 virtual int cms::MapMessage::getInt (const std::string & *name*) const [pure virtual]

Returns the Int value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 213).

6.333.3.8 virtual long long cms::MapMessage::getLong (const std::string & *name*) const [pure virtual]

Returns the Long value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 214).

6.333.3.9 virtual std::vector< std::string > cms::MapMessage::getMapNames ( ) const [pure virtual]

Returns an Enumeration of all the names in the **MapMessage** (p. 1379) object.

Returns

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 1379)

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 214).

6.333.3.10 virtual short cms::MapMessage::getShort (const std::string & *name*) const [pure virtual]

Returns the Short value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 215).

6.333.3.11 virtual std::string cms::MapMessage::getString (const std::string & *name*) const [pure virtual]

Returns the String value of the Specified name.

Parameters

name	Name of the value to fetch from the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 215).

6.333.3.12 virtual bool cms::MapMessage::isEmpty ( ) const [pure virtual]

Returns true if there are no values stored in the **MapMessage** (p. 1379) body.

Returns

true if the body of the **MapMessage** (p. 1379) contains no elements.

Exceptions

CMSException *(p. 640)*	if the operation fails due to an internal error.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 215).

6.333.3.13 virtual bool cms::MapMessage::itemExists (const std::string & *name*) const [pure virtual]

Indicates whether an item exists in this **MapMessage** (p. 1379) object.

Parameters

name	String name of the Object in question

Returns

boolean value indicating if the name is in the map

Exceptions

CMSException *(p. 640)*	- if the operation fails due to an internal error.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 216).

6.333.3.14 virtual void cms::MapMessage::setBoolean (const std::string & *name*, bool *value*) [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

CMSException *(p. 640)*	- if the operation fails due to an internal error.
MessageNotWritable-Exception	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 216).

6.333.3.15 virtual void cms::MapMessage::setByte (const std::string & *name,* unsigned char *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 216).

6.333.3.16 virtual void cms::MapMessage::setBytes (const std::string & *name,* const std::vector< unsigned char > & *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 217).

6.333.3.17 virtual void cms::MapMessage::setChar (const std::string & *name,* char *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 217).

6.333.3.18 virtual void cms::MapMessage::setDouble (const std::string & *name*, double *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 217).

6.333.3.19 virtual void cms::MapMessage::setFloat (const std::string & *name*, float *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 218).

6.333.3.20 virtual void cms::MapMessage::setInt (const std::string & *name*, int *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 218).

6.333.3.21 virtual void cms::MapMessage::setLong (const std::string & *name,* long long *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 218).

6.333.3.22 virtual void cms::MapMessage::setShort (const std::string & *name,* short *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 219).

6.333.3.23 virtual void cms::MapMessage::setString (const std::string & *name,* const std::string & *value*) [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. 640)*	- if the operation fails due to an internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the **Message** (p. 1426) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 219).

The documentation for this class was generated from the following file:

- src/main/cms/**MapMessage.h**

6.334 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** ()

6.334.1 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.

6.334.2 Constructor & Destructor Documentation

6.334.2.1 decaf::util::logging::MarkBlockLogger::MarkBlockLogger (Logger * *logger,* const std::string & *blockName*) [inline]

Constructor - Marks Block entry.

Parameters

logger	**Logger** (p. 1312) to use
blockName	Block name

6.334.2.2 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**

6.335 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

  Called before marshaling is started to prepare the object to be marshaled.
- virtual void **afterMarshal** (**WireFormat** *wireFormat)=0

  Called after marshaling is started to cleanup the object being marshaled.
- virtual void **beforeUnmarshal** (**WireFormat** *wireFormat)=0

  Called before unmarshaling is started to prepare the object to be unmarshaled.
- virtual void **afterUnmarshal** (**WireFormat** *wireFormat)=0

  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.

6.335.1 Constructor & Destructor Documentation

6.335.1.1 virtual activemq::wireformat::MarshalAware::~MarshalAware ( ) [inline, virtual]

6.335.2 Member Function Documentation

6.335.2.1 virtual void activemq::wireformat::MarshalAware::afterMarshal (WireFormat * *wireFormat*) [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

Parameters

wireFormat	The wireformat object to control marshaling

Exceptions

IOException	if an I/O error occurs.

6.335.2.2 virtual void activemq::wireformat::MarshalAware::afterUnmarshal (WireFormat * *wireFormat*) [pure virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

wireFormat	The wireformat object to control marshaling

Exceptions

IOException	if an I/O error occurs.

6.335.2.3 virtual void activemq::wireformat::MarshalAware::beforeMarshal (WireFormat * *wireFormat*) [pure virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

wireFormat	The wireformat object to control marshaling

Exceptions

IOException	if an I/O error occurs.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 210), and **activemq::commands::ActiveMQTextMessage** (p. 315).

6.335.2.4 virtual void activemq::wireformat::MarshalAware::beforeUnmarshal (WireFormat * *wireFormat*) [pure virtual]

Called before unmarshaling is started to prepare the object to be unmarshaled.

Parameters

wireFormat	The wireformat object to control marshaling

Exceptions

IOException	if an I/O error occurs.

6.335.2.5 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.

6.335.2.6 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::Message** (p. 1421), **activemq::commands::WireFormatInfo** (p. 2244), **activemq::commands::ActiveMQMapMessage** (p. 216), and **activemq::commands::BaseDataStructure** (p. 410).

6.335.2.7 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
wireFormat	The wireformat object to control marshaling
data	A vector of bytes that contains the object in marshaled form.

Exceptions

IOException	if an I/O error occurs.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**MarshalAware.h**

6.336 activemq::wireformat::openwire::marshal::generated::MarshallerFactory Class Reference

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

`#include <src/main/activemq/wireformat/openwire/marshal/generated/MarshallerFactory.h>`

Public Member Functions

- virtual ~**MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** *format)

6.336.1 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

6.336.2 Constructor & Destructor Documentation

6.336.2.1 virtual activemq::wireformat::openwire::marshal::generated::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]

6.336.3 Member Function Documentation

6.336.3.1 virtual void activemq::wireformat::openwire::marshal::generated::MarshallerFactory::configure (OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MarshallerFactory.h**

6.337 activemq::util::MarshallingSupport Class Reference

`#include <src/main/activemq/util/MarshallingSupport.h>`

Public Member Functions

- **MarshallingSupport** ()
- virtual ~**MarshallingSupport** ()

Static Public Member Functions

- static void **writeString** (**decaf::io::DataOutputStream** &dataOut, **const** std::string &value)

  Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static void **writeString16** (**decaf::io::DataOutputStream** &dataOut, **const** std::string &value)

  Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static void **writeString32** (**decaf::io::DataOutputStream** &dataOut, **const** std::string &value)

  Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.
- static std::string **readString16** (**decaf::io::DataInputStream** &dataIn)

  Reads an Openwire encoded string from the provided DataInputStream.
- static std::string **readString32** (**decaf::io::DataInputStream** &dataIn)

  Reads an Openwire encoded string from the provided DataInputStream.
- static std::string **asciiToModifiedUtf8** (**const** std::string &asciiString)

  Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.
- static std::string **modifiedUtf8ToAscii** (**const** std::string modifiedUtf8String)

  Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

6.337.1 Constructor & Destructor Documentation

6.337.1.1 activemq::util::MarshallingSupport::MarshallingSupport ( )

6.337.1.2 virtual activemq::util::MarshallingSupport::~MarshallingSupport ( ) [virtual]

6.337.2 Member Function Documentation

6.337.2.1 static std::string activemq::util::MarshallingSupport::asciiToModifiedUtf8 (const std::string & *asciiString*) [static]

Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.

This allows an ASCII string containing values greater than 127 as well as embedded NULLs to be sent to a Java client.

Parameters

asciiString	The ASCII string to encode as Modified UTF-8

Returns

a string containing the Modified UTF-8 encoded form of the provided string.

Exceptions

UTFDataFormatException	if the length of the encoded string would exceed the size of an signed integer.

6.337.2.2 static std::string activemq::util::MarshallingSupport::modifiedUtf8ToAscii (const std::string *modifiedUtf8String*) [static]

Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

This will handle any string sent from a Java client which contains values within the [0..255] range or has embedded Nulls. Strings that have encoded values greater than 255 will cause an exception to be thrown.

Parameters

modifiedUtf8-String	The string to convert from Modified UTF-8 to ASCII.

Returns

the ASCII encoded version of the provided string.

Exceptions

UTFDataFormatException	if the provided string contains invalid data or the character values encoded in the string exceed ASCII value 255.

6.337.2.3 static std::string activemq::util::MarshallingSupport::readString16 (decaf::io::DataInputStream & *dataIn*) [static]

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 16bits.

Parameters

dataIn	The DataInputStream to read the String data from.

Returns

the String value.

Exceptions

IOException	if an I/O error occurs while writing the string.

6.337.2.4 static std::string activemq::util::MarshallingSupport::readString32 (decaf::io::DataInputStream & *dataIn*) [static]

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 32bits.

Parameters

dataIn	The DataInputStream to read the String data from.

Returns

the String value.

Exceptions

IOException	if an I/O error occurs while writing the string.

6.337.2.5 static void activemq::util::MarshallingSupport::writeString (decaf::io::DataOutputStream & *dataOut,* const std::string & *value*) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed.

Parameters

dataOut	The DataOutputStream to write the String data to.
value	Thre String value to write in Openwire form.

Exceptions

IOException	if an I/O error occurs while writing the string.

6.337.2.6 static void activemq::util::MarshallingSupport::writeString16 (decaf::io::DataOutputStream & *dataOut,* const std::string & *value*) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a short and the string data no Openwire Type tag is appended.

Parameters

dataOut	The DataOutputStream to write the String data to.
value	Thre String value to write in Openwire form.

Exceptions

IOException	if an I/O error occurs while writing the string.

6.337.2.7 static void activemq::util::MarshallingSupport::writeString32 (decaf::io::DataOutputStream & *dataOut,* const std::string & *value*) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a int and the string data no Openwire Type tag is appended.

Parameters

dataOut	The DataOutputStream to write the String data to.
value	Thre String value to write in Openwire form.

Exceptions

IOException	if an I/O error occurs while writing the string.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MarshallingSupport.h**

6.338 decaf::lang::Math Class Reference

The class **Math** (p. 1396) 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 unsigned 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**

6.338.1 Detailed Description

The class **`Math`** (p. 1396) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

6.338.2 Constructor & Destructor Documentation

6.338.2.1 decaf::lang::Math::Math ( ) [inline]

6.338.2.2 virtual decaf::lang::Math::~Math ( ) [inline, virtual]

6.338.3 Member Function Documentation

6.338.3.1 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

6.338.3.2 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

6.338.3.3 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. 1045)(0x7fffffff & Float::floatToIntBits(value))

Parameters

value	- the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.338.3.4 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. 962)(0x7fffffffffffffffULL & Double::doubleToLongBits(value))

Parameters

value	- the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.338.3.5 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 10n 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 log1p(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.

6.338.3.6 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.

6.338.3.7 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. 1866). 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`.

6.338.3.8 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. 1174). 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`.

6.338.3.9 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. 1351). 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`.

6.338.3.10 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`.

6.338.3.11 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`.

6.338.3.12 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. 1866). 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`.

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::drainTo().

6.338.3.13 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. 1175). 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`.

6.338.3.14 static unsigned 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. 1175). 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`.

6.338.3.15 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. 1351). 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`.

6.338.3.16 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`.

6.338.3.17 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`.

6.338.3.18 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.

6.338.3.19 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 × 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.

6.338.3.20 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. 1400)(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. 1175), the result is equal to the value of **Integer::MIN_VALUE** (p. 1175). o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX_VALUE** (p. 1174), the result is equal to the value of **Integer::MAX_VALUE** (p. 1174).

Parameters

value	- the value to round

Returns

the value of the argument rounded to the nearest integral value.

6.338.3.21 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. 1400)(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. 1351), the result is equal to the value of **Long::MIN_VALUE** (p. 1351). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX_VALUE** (p. 1351), the result is equal to the value of **Long::MAX_VALUE** (p. 1351).

Parameters

value	- the value to round

Returns

the value of the argument rounded to the nearest integral value.

6.338.3.22 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^a, 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 ex 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 ex - 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, 1.0f 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

6.338.3.23 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

6.338.3.24 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 (ex + 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 (ex - 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 (ex - e-x)/(ex + 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.

6.338.3.25 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.

6.338.3.26 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.

6.338.4 Field Documentation

6.338.4.1 const double decaf::lang::Math::E [static]

6.338.4.2 const double decaf::lang::Math::PI [static]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Math.h**

6.339 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. 2226) monitor with a set limit.*
- virtual ~**MemoryUsage** ()
- virtual void **waitForSpace** ()

  *Waits forever for more space to be returned to this **Usage** (p. 2226) Manager.*
- virtual void **waitForSpace** (unsigned int timeout)

  *Waits for more space to be returned to this **Usage** (p. 2226) 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. 2226) 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.

6.339.1 Constructor & Destructor Documentation

6.339.1.1 activemq::util::MemoryUsage::MemoryUsage ( )

Default Constructor.

6.339.1.2 activemq::util::MemoryUsage::MemoryUsage (unsigned long long *limit*)

Creates an instance of an **Usage** (p. 2226) monitor with a set limit.

Parameters

limit	- amount of memory this manager allows.

6.339.1.3 virtual activemq::util::MemoryUsage::~MemoryUsage ( ) [virtual]

6.339.2 Member Function Documentation

6.339.2.1 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. 2227).

6.339.2.2 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. 2227).

6.339.2.3 unsigned long long activemq::util::MemoryUsage::getLimit ( ) const [inline]

Gets the current limit amount.

Returns

the amount that can be used before full.

6.339.2.4 unsigned long long activemq::util::MemoryUsage::getUsage ( ) const [inline]

Gets the current usage amount.

Returns

the amount of bytes currently used.

6.339.2.5 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. 2227).

6.339.2.6 virtual bool activemq::util::MemoryUsage::isFull ( ) const [virtual]

Returns true if this **Usage** (p. 2226) instance is full, i.e.

Usage (p. 2226) >= 100%

Implements **activemq::util::Usage** (p. 2227).

6.339.2.7 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.

6.339.2.8 void activemq::util::MemoryUsage::setUsage (unsigned long long *usage*) [inline]

Sets the current usage amount.

Parameters

usage	- The amount to tag as used.

6.339.2.9 virtual void activemq::util::MemoryUsage::waitForSpace ( ) [virtual]

Waits forever for more space to be returned to this **Usage** (p. 2226) Manager.

Implements **activemq::util::Usage** (p. 2227).

6.339.2.10 virtual void activemq::util::MemoryUsage::waitForSpace (unsigned int *timeout*) [virtual]

Waits for more space to be returned to this **Usage** (p. 2226) 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. 2228).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MemoryUsage.h**

6.340 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **Message** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** ∗wireFormat AMQCPP_UNUSED)

  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)

  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. 1412) will use when the Acknowledge method is called.*
- virtual **Pointer**
  < **core::ActiveMQAckHandler** > **getAckHandler** () **const**

  *Gets the Acknowledgment Handler that this **Message** (p. 1412) will use when the Acknowledge method is called.*
- void **setConnection** (**core::ActiveMQConnection** ∗**connection**)

*Sets the ActiveMQConnection instance that this **Command** (p. 671) was created from when the session create methods are called to create a **Message** (p. 1412).*

- **core::ActiveMQConnection** * **getConnection** () **const**

  *Gets the ActiveMQConnection instance that this **Command** (p. 671) was created from when the session create methods are called to create a **Message** (p. 1412).*

- 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. 1412) classes to perform tasks before a message is sent.*

- **util::PrimitiveMap** & **getMessageProperties** ()

  *Gets a reference to the **Message** (p. 1412)'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. 1412) Properties Are Read Only.*

- void **setReadOnlyProperties** (bool value)

  *Set the Read Only State of the **Message** (p. 1412) Properties.*

- bool **isReadOnlyBody** () **const**

  *Returns if the **Message** (p. 1412) Body is Read Only.*

- void **setReadOnlyBody** (bool value)

  *Set the Read Only State of the **Message** (p. 1412) 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 > & **getMarshalledProperties** () **const**
- virtual std::vector< unsigned
  char > & **getMarshalledProperties** ()
- virtual void **setMarshalledProperties** (**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)

  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 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**
- **core::ActiveMQConnection** * **connection**

Static Protected Attributes

- static **const** unsigned int **DEFAULT_MESSAGE_SIZE** = 1024

6.340.1 Constructor & Destructor Documentation

6.340.1.1 activemq::commands::Message::Message ( )

6.340.1.2 virtual activemq::commands::Message::~Message ( ) [virtual]

6.340.2 Member Function Documentation

6.340.2.1 virtual void activemq::commands::Message::afterUnmarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) [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. 410).

6.340.2.2 virtual void activemq::commands::Message::beforeMarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) [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. 410).

6.340.2.3 virtual Message* activemq::commands::Message::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::ActiveMQBlobMessage** (p. 123), **activemq::commands::ActiveMQTextMessage** (p. 315), **activemq::commands::ActiveMQMessage** (p. 224), and **activemq::commands::ActiveMQObjectMessage** (p. 234).

6.340.2.4 virtual void activemq::commands::Message::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. 382).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::ActiveMQBlobMessage** (p. 123), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 224).

6.340.2.5 virtual bool activemq::commands::Message::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 132), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230), **activemq::commands::ActiveMQBlobMessage** (p. 124), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 224).

6.340.2.6 virtual Pointer<core::ActiveMQAckHandler> activemq::commands::Message::getAckHandler ( ) const [inline, virtual]

Gets the Acknowledgment Handler that this **Message** (p. 1412) will use when the Acknowledge method is called.

Returns

handler ActiveMQAckHandler to call or NULL if not set

6.340.2.7 virtual long long activemq::commands::Message::getArrival ( ) const [virtual]

6.340.2.8 virtual long long activemq::commands::Message::getBrokerInTime ( ) const [virtual]

6.340.2.9 virtual long long activemq::commands::Message::getBrokerOutTime ( ) const [virtual]

6.340.2.10 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath ( ) const [virtual]

6.340.2.11 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath ( ) [virtual]

6.340.2.12 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster ( ) const [virtual]

6.340.2.13 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster ( ) [virtual]

6.340.2.14 core::ActiveMQConnection* activemq::commands::Message::getConnection ( ) const [inline]

Gets the ActiveMQConnection instance that this **Command** (p. 671) was created from when the session create methods are called to create a **Message** (p. 1412).

Returns

the ActiveMQConnection parent for this **Message** (p. 1412) or NULL if not set.

6.340.2.15 virtual const std::vector<unsigned char>& activemq::commands::Message::getContent ( ) const [virtual]

6.340.2.16 virtual std::vector<unsigned char>& activemq::commands::Message::getContent ( ) [virtual]

6.340.2.17 virtual const std::string& activemq::commands::Message::getCorrelationId ( ) const [virtual]

6.340.2.18 virtual std::string& activemq::commands::Message::getCorrelationId ( ) [virtual]

6.340.2.19 virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure ( ) const [virtual]

6.340.2.20 virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure ( ) [virtual]

6.340.2.21 virtual unsigned char activemq::commands::Message::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 132), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ActiveMQMapMessage** (p. 213), **activemq::commands::ActiveMQBlobMessage** (p. 124), **activemq::commands::ActiveMQTextMessage** (p. 316), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 225).

6.340.2.22 virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ( ) const [virtual]

6.340.2.23 virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ( ) [virtual]

6.340.2.24 virtual long long activemq::commands::Message::getExpiration ( ) const [virtual]

6.340.2.25 virtual const std::string& activemq::commands::Message::getGroupID ( ) const [virtual]

6.340.2.26 virtual std::string& activemq::commands::Message::getGroupID ( ) [virtual]

6.340.2.27 virtual int activemq::commands::Message::getGroupSequence ( ) const [virtual]

6.340.2.28 virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshalledProperties ( ) const [virtual]

6.340.2.29 virtual std::vector<unsigned char>& activemq::commands::Message::getMarshalledProperties ( ) [virtual]

6.340.2.30 virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId ( ) const [virtual]

6.340.2.31 virtual Pointer<MessageId>& activemq::commands::Message::getMessageId ( ) [virtual]

6.340.2.32 util::PrimitiveMap& activemq::commands::Message::getMessageProperties ( ) [inline]

Gets a reference to the **Message** (p. 1412)'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.

6.340.2.33 const util::PrimitiveMap& activemq::commands::Message::getMessageProperties ( ) const [inline]

6.340.2.34 virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ( ) const [virtual]

6.340.2.35 virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ( ) [virtual]

6.340.2.36 virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ( ) const [virtual]

6.340.2.37 virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ( ) [virtual]

6.340.2.38 virtual unsigned char activemq::commands::Message::getPriority ( ) const [virtual]

6.340.2.39 virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId ( ) const [virtual]

6.340.2.40 virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId ( ) [virtual]

6.340.2.41 virtual int activemq::commands::Message::getRedeliveryCounter ( ) const [virtual]

6.340.2.42 virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ( ) const [virtual]

6.340.2.43 virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ( ) [virtual]

6.340.2.44 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. 316).

6.340.2.45 virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ( ) const [virtual]

6.340.2.46 virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ( ) [virtual]

6.340.2.47 virtual long long activemq::commands::Message::getTimestamp ( ) const [virtual]

6.340.2.48 virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId ( ) const [virtual]

6.340.2.49 virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId ( ) [virtual]

6.340.2.50 virtual const std::string& activemq::commands::Message::getType ( ) const [virtual]

6.340.2.51 virtual std::string& activemq::commands::Message::getType ( ) [virtual]

6.340.2.52 virtual const std::string& activemq::commands::Message::getUserID ( ) const [virtual]

6.340.2.53 virtual std::string& activemq::commands::Message::getUserID ( ) [virtual]

6.340.2.54 virtual bool activemq::commands::Message::isCompressed ( ) const [virtual]

6.340.2.55 virtual bool activemq::commands::Message::isDroppable ( ) const [virtual]

6.340.2.56 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.

6.340.2.57 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. 410).

Reimplemented in **activemq::commands::ActiveMQMapMessage** (p. 216).

6.340.2.58 virtual bool activemq::commands::Message::isMessage ( ) const [inline, virtual]

Returns

an answer of true to the **isMessage()** (p. 1421) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.340.2.59 virtual bool activemq::commands::Message::isPersistent ( ) const [virtual]

6.340.2.60 bool activemq::commands::Message::isReadOnlyBody ( ) const [inline]

Returns if the **Message** (p. 1412) Body is Read Only.

Returns

true if **Message** (p. 1412) Content is Read Only.

6.340.2.61 bool activemq::commands::Message::isReadOnlyProperties ( ) const [inline]

Returns if the **Message** (p. 1412) Properties Are Read Only.

Returns

true if **Message** (p. 1412) Properties are Read Only.

6.340.2.62 virtual bool activemq::commands::Message::isRecievedByDFBridge ( ) const [virtual]

6.340.2.63 virtual void activemq::commands::Message::onSend ( ) [inline, virtual]

Allows derived **Message** (p. 1412) classes to perform tasks before a message is sent.

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 133), **activemq::commands::ActiveMQStreamMessage** (p. 282), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.340.2.64 virtual void activemq::commands::Message::setAckHandler (const Pointer< core::ActiveMQAckHandler > & *handler*) [inline, virtual]

Sets the Acknowledgment Handler that this **Message** (p. 1412) will use when the Acknowledge method is called.

Parameters

handler	ActiveMQAckHandler to call

6.340.2.65 virtual void activemq::commands::Message::setArrival (long long *arrival*) [virtual]

6.340.2.66 virtual void activemq::commands::Message::setBrokerInTime (long long *brokerInTime*) [virtual]

6.340.2.67 virtual void **activemq::commands::Message::setBrokerOutTime** (long long *brokerOutTime*) [virtual]

6.340.2.68 virtual void **activemq::commands::Message::setBrokerPath** (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.340.2.69 virtual void **activemq::commands::Message::setCluster** (const std::vector< decaf::lang::Pointer< BrokerId > > & *cluster*) [virtual]

6.340.2.70 virtual void **activemq::commands::Message::setCompressed** (bool *compressed*) [virtual]

6.340.2.71 void **activemq::commands::Message::setConnection** (core::ActiveMQConnection * *connection*) [inline]

Sets the ActiveMQConnection instance that this **Command** (p. 671) was created from when the session create methods are called to create a **Message** (p. 1412).

Parameters

handler	ActiveMQConnection parent for this message

6.340.2.72 virtual void **activemq::commands::Message::setContent** (const std::vector< unsigned char > & *content*) [virtual]

6.340.2.73 virtual void **activemq::commands::Message::setCorrelationId** (const std::string & *correlationId*) [virtual]

6.340.2.74 virtual void **activemq::commands::Message::setDataStructure** (const Pointer< DataStructure > & *dataStructure*) [virtual]

6.340.2.75 virtual void **activemq::commands::Message::setDestination** (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.340.2.76 virtual void **activemq::commands::Message::setDroppable** (bool *droppable*) [virtual]

6.340.2.77 virtual void **activemq::commands::Message::setExpiration** (long long *expiration*) [virtual]

6.340.2.78 virtual void **activemq::commands::Message::setGroupID** (const std::string & *groupID*) [virtual]

6.340.2.79 virtual void **activemq::commands::Message::setGroupSequence** (int *groupSequence*) [virtual]

6.340.2.80 virtual void **activemq::commands::Message::setMarshalledProperties** (const std::vector< unsigned char > & *marshalledProperties*) [virtual]

6.340.2.81 virtual void **activemq::commands::Message::setMessageId** (const Pointer< MessageId > & *messageId*) [virtual]

6.340.2.82 virtual void **activemq::commands::Message::setOriginalDestination** (const Pointer< ActiveMQDestination > & *originalDestination*) [virtual]

6.340.2.83 virtual void **activemq::commands::Message::setOriginalTransactionId** (const Pointer< TransactionId > & *originalTransactionId*) [virtual]

6.340.2.84 virtual void **activemq::commands::Message::setPersistent** (bool *persistent*) [virtual]

6.340.2.85 virtual void **activemq::commands::Message::setPriority** (unsigned char *priority*) [virtual]

6.340.2.86 virtual void **activemq::commands::Message::setProducerId** (const Pointer< ProducerId > & *producerId*) [virtual]

6.340.2.87 void **activemq::commands::Message::setReadOnlyBody** (bool *value*) [inline]

Set the Read Only State of the **Message** (p. 1412) Content.

Parameters

value	- true if Content should be read only.

6.340.2.88 void **activemq::commands::Message::setReadOnlyProperties** (bool *value*) [inline]

Set the Read Only State of the **Message** (p. 1412) Properties.

Parameters

value	- true if Properties should be read only.

6.340.2.89 virtual void **activemq::commands::Message::setRecievedByDFBridge** (bool *recievedByDFBridge*) [virtual]

6.340.2.90 virtual void **activemq::commands::Message::setRedeliveryCounter** (int *redeliveryCounter*) [virtual]

6.340.2.91 virtual void **activemq::commands::Message::setReplyTo** (const Pointer< ActiveMQDestination > & *replyTo*) [virtual]

6.340.2.92 virtual void **activemq::commands::Message::setTargetConsumerId** (const Pointer< ConsumerId > & *targetConsumerId*) [virtual]

6.340.2.93 virtual void **activemq::commands::Message::setTimestamp** (long long *timestamp*) [virtual]

6.340.2.94 virtual void **activemq::commands::Message::setTransactionId** (const Pointer< TransactionId > & *transactionId*) [virtual]

6.340.2.95 virtual void **activemq::commands::Message::setType** (const std::string & *type*) [virtual]

6.340.2.96 virtual void **activemq::commands::Message::setUserID** (const std::string & *userID*) [virtual]

6.340.2.97 virtual std::string **activemq::commands::Message::toString** ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 138), **activemq::commands::ActiveMQStreamMessage** (p. 286), **activemq::commands::ActiveMQMapMessage** (p. 219), **activemq::commands::ActiveMQBlobMessage** (p. 125), **activemq::commands::ActiveMQTextMessage** (p. 317), **activemq::commands::ActiveMQObjectMessage** (p. 235), and **activemq::commands::ActiveMQMessage** (p. 225).

6.340.2.98 virtual Pointer<Command> activemq::commands::Message::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.340.3 Field Documentation

6.340.3.1 long long activemq::commands::Message::arrival [protected]

6.340.3.2 long long activemq::commands::Message::brokerInTime [protected]

6.340.3.3 long long activemq::commands::Message::brokerOutTime [protected]

6.340.3.4 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::brokerPath [protected]

6.340.3.5 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::cluster [protected]

6.340.3.6 bool activemq::commands::Message::compressed [protected]

6.340.3.7 core::ActiveMQConnection* activemq::commands::Message::connection [protected]

6.340.3.8 std::vector<unsigned char> activemq::commands::Message::content [protected]

6.340.3.9 std::string activemq::commands::Message::correlationId [protected]

6.340.3.10 Pointer<DataStructure> activemq::commands::Message::dataStructure [protected]

6.340.3.11 const unsigned int activemq::commands::Message::DEFAULT_MESSAGE_SIZE = 1024 [static, protected]

6.340.3.12 Pointer<ActiveMQDestination> activemq::commands::Message::destination [protected]

6.340.3.13 bool activemq::commands::Message::droppable [protected]

6.340.3.14 long long activemq::commands::Message::expiration [protected]

6.340.3.15 std::string activemq::commands::Message::groupID [protected]

6.340.3.16 int activemq::commands::Message::groupSequence [protected]

6.340.3.17 const unsigned char activemq::commands::Message::ID_MESSAGE = 0 [static]

6.340.3.18 std::vector<unsigned char> activemq::commands::Message::marshalledProperties [protected]

6.340.3.19 Pointer<MessageId> activemq::commands::Message::messageId [protected]

6.340.3.20 Pointer<ActiveMQDestination> activemq::commands::Message::originalDestination [protected]

6.340.3.21 Pointer<TransactionId> activemq::commands::Message::originalTransactionId [protected]

6.340.3.22 bool activemq::commands::Message::persistent [protected]

6.340.3.23 unsigned char activemq::commands::Message::priority [protected]

6.340.3.24 Pointer<ProducerId> activemq::commands::Message::producerId [protected]

6.340.3.25 bool activemq::commands::Message::recievedByDFBridge [protected]

6.340.3.26 int activemq::commands::Message::redeliveryCounter [protected]

6.340.3.27 Pointer<ActiveMQDestination> activemq::commands::Message::replyTo [protected]

6.340.3.28 Pointer<ConsumerId> activemq::commands::Message::targetConsumerId [protected]

6.340.3.29 long long activemq::commands::Message::timestamp [protected]

6.340.3.30 Pointer<TransactionId> activemq::commands::Message::transactionId [protected]

6.340.3.31 std::string activemq::commands::Message::type [protected]

6.340.3.32 std::string activemq::commands::Message::userID [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Message.h**

6.341 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

  Acknowledges all consumed messages of the session of this consumed message.
- virtual void **clearBody** ()=0

  Clears out the body of the message.
- virtual void **clearProperties** ()=0

  Clears out the message body.
- virtual std::vector< std::string > **getPropertyNames** () **const** =0

Retrieves the property names.

- virtual bool **propertyExists** (**const** std::string &name) **const** =0

  Indicates whether or not a given property exists.
- virtual bool **getBooleanProperty** (**const** std::string &name) **const** =0

  Gets a boolean property.
- virtual unsigned char **getByteProperty** (**const** std::string &name) **const** =0

  Gets a byte property.
- virtual double **getDoubleProperty** (**const** std::string &name) **const** =0

  Gets a double property.
- virtual float **getFloatProperty** (**const** std::string &name) **const** =0

  Gets a float property.
- virtual int **getIntProperty** (**const** std::string &name) **const** =0

  Gets a int property.
- virtual long long **getLongProperty** (**const** std::string &name) **const** =0

  Gets a long property.
- virtual short **getShortProperty** (**const** std::string &name) **const** =0

  Gets a short property.
- virtual std::string **getStringProperty** (**const** std::string &name) **const** =0

  Gets a string property.
- virtual void **setBooleanProperty** (**const** std::string &name, bool value)=0

  Sets a boolean property.
- virtual void **setByteProperty** (**const** std::string &name, unsigned char value)=0

  Sets a byte property.
- virtual void **setDoubleProperty** (**const** std::string &name, double value)=0

  Sets a double property.
- virtual void **setFloatProperty** (**const** std::string &name, float value)=0

  Sets a float property.
- virtual void **setIntProperty** (**const** std::string &name, int value)=0

  Sets a int property.
- virtual void **setLongProperty** (**const** std::string &name, long long value)=0

  Sets a long property.
- virtual void **setShortProperty** (**const** std::string &name, short value)=0

  Sets a short property.
- virtual void **setStringProperty** (**const** std::string &name, **const** std::string &value)=0

  Sets a string property.
- virtual std::string **getCMSCorrelationID** () **const** =0

  Gets the correlation ID for the message.
- virtual void **setCMSCorrelationID** (**const** std::string &correlationId)=0

  Sets the correlation ID for the message.
- virtual int **getCMSDeliveryMode** () **const** =0

  *Gets the **DeliveryMode** (p. 925) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0

  *Sets the **DeliveryMode** (p. 925) for this message.*
- virtual **const Destination** ∗ **getCMSDestination** () **const** =0

  *Gets the **Destination** (p. 936) object for this message.*
- virtual void **setCMSDestination** (**const Destination** ∗destination)=0

  *Sets the **Destination** (p. 936) object for this message.*
- virtual long long **getCMSExpiration** () **const** =0

  Gets the message's expiration value.
- virtual void **setCMSExpiration** (long long expireTime)=0

  Sets the message's expiration value.

- virtual std::string **getCMSMessageID** () **const** =0

  The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.
- virtual void **setCMSMessageID** (**const** std::string &id)=0

  Sets the message ID.
- virtual int **getCMSPriority** () **const** =0

  Gets the message priority level.
- virtual void **setCMSPriority** (int priority)=0

  Sets the Priority Value for this message.
- virtual bool **getCMSRedelivered** () **const** =0

  Gets an indication of whether this message is being redelivered.
- virtual void **setCMSRedelivered** (bool redelivered)=0

  Specifies whether this message is being redelivered.
- virtual **const cms::Destination** ∗ **getCMSReplyTo** () **const** =0

  *Gets the **Destination** (p. 936) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (**const cms::Destination** ∗destination)=0

  *Sets the **Destination** (p. 936) object to which a reply to this message should be sent.*
- virtual long long **getCMSTimestamp** () **const** =0

  Gets the message timestamp.
- virtual void **setCMSTimestamp** (long long timeStamp)=0

  Sets the message timestamp.
- virtual std::string **getCMSType** () **const** =0

  Gets the message type identifier supplied by the client when the message was sent.
- virtual void **setCMSType** (**const** std::string &type)=0

  Sets the message type.

Static Public Attributes

- static **const** int **DEFAULT_DELIVERY_MODE**

  *The Default delivery mode for **Message** (p. 1426) Producers is PERSISTENT.*
- static **const** int **DEFAULT_MSG_PRIORITY**

  *The Default priority assigned to a **Message** (p. 1426) is 4.*
- static **const** long long **DEFAULT_TIME_TO_LIVE**

  *The Default Time to Live for a **Message** (p. 1426) Producer is unlimited, the message will never expire.*

6.341.1 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. 1426) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 1426) Interface definition.

- Stream - A **StreamMessage** (p. 2020) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 557) type the values written to a **StreamMessage** (p. 2020) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 1426) Body.
  - Map - A **MapMessage** (p. 1379) 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. 1379) makes no garuntee on the order of the elements within the **Message** (p. 1426) body.

- Text - A **TextMessage** (p. 2093) 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. 557) 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. 1426) Properties

Message (p. 1426) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 640). 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. 1426) 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** (p. 1426)'s properties will result in a **CMSException** (p. 640) being thrown.

See also

JMS API

Since

1.0

6.341.2 Constructor & Destructor Documentation

6.341.2.1 virtual cms::Message::~Message ( ) [virtual]

6.341.3 Member Function Documentation

6.341.3.1 virtual void cms::Message::acknowledge ( ) const [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

CMSException *(p. 640)*	- if an internal error occurs.
IllegalStateException *(p. 1098)*	- if this method is called on a closed session.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230).

6.341.3.2 virtual void cms::Message::clearBody ( ) [pure virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230), **activemq::commands::ActiveMQStreamMessage** (p. 280), **activemq::commands::ActiveMQMapMessage** (p. 210), and **activemq::commands::ActiveMQTextMessage** (p. 315).

6.341.3.3 virtual void cms::Message::clearProperties ( ) [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230).

6.341.3.4 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. 559), **activemq::commands::ActiveMQBytesMessage** (p. 131), **activemq::commands::ActiveMQStreamMessage** (p. 281), **activemq::commands::ActiveMQMapMessage** (p. 211), **activemq::commands::ActiveMQBlobMessage** (p. 123), **activemq::commands::ActiveMQTextMessage** (p. 315), **activemq::commands::ActiveMQObjectMessage** (p. 234), and **activemq::commands::ActiveMQMessage** (p. 224).

6.341.3.5 virtual bool cms::Message::getBooleanProperty (const std::string & *name*) const [pure virtual]

Gets a boolean property.

Parameters

name	The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException *(p. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230).

6.341.3.6 virtual unsigned char cms::Message::getByteProperty (const std::string & *name*) const [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. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 230), **activemq::commands::**

::ActiveMQMessageTemplate< cms::Message > (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 230), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 230), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 230).

6.341.3.7 virtual std::string cms::Message::getCMSCorrelationID ( ) const [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. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.8 virtual int cms::Message::getCMSDeliveryMode ( ) const [pure virtual]

Gets the **DeliveryMode** (p. 925) for this message.

Returns

DeliveryMode (p. 925) enumerated value.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.9 virtual const Destination* cms::Message::getCMSDestination ( ) const [pure virtual]

Gets the **Destination** (p. 936) 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. 936) object

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.10 virtual long long cms::Message::getCMSExpiration ( ) const [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. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.11 virtual std::string cms::Message::getCMSMessageID ( ) const [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. 1496) 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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.12 virtual int cms::Message::getCMSPriority ( ) const [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.13 virtual bool cms::Message::getCMSRedelivered ( ) const [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.14 virtual const cms::Destination* cms::Message::getCMSReplyTo ( ) const [pure virtual]

Gets the **Destination** (p. 936) object to which a reply to this message should be sent.

Returns

Destination (p. 936) to which to send a response to this message

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.15 virtual long long cms::Message::getCMSTimestamp ( ) const [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands-**

::ActiveMQMessageTemplate< cms::Message > (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.16 virtual std::string cms::Message::getCMSType ( ) const [pure virtual]

Gets the message type identifier supplied by the client when the message was sent.

Returns

the message type

See also

setCMSType (p. 1444)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.17 virtual double cms::Message::getDoubleProperty (const std::string & *name*) const [pure virtual]

Gets a double property.

Parameters

name	The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException *(p. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.18 virtual float cms::Message::getFloatProperty (const std::string & *name*) const [pure virtual]

Gets a float property.

Parameters

name	The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException *(p. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.19 virtual int cms::Message::getIntProperty (const std::string & *name*) const [pure virtual]

Gets a int property.

Parameters

name	The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException *(p. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.20 virtual long long cms::Message::getLongProperty (const std::string & *name*) const [pure virtual]

Gets a long property.

Parameters

name	The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException *(p. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.21 virtual std::vector<std::string> cms::Message::getPropertyNames ( ) const [pure virtual]

Retrieves the property names.

Returns

The complete set of property names currently in this message.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.22 virtual short cms::Message::getShortProperty (const std::string & *name*) const [pure virtual]

Gets a short property.

Parameters

name	The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException *(p. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.23 virtual std::string cms::Message::getStringProperty (const std::string & *name*) const [pure virtual]

Gets a string property.

Parameters

name	The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException *(p. 640)*	if the property does not exist.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 231), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 231), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 231).

6.341.3.24 virtual bool cms::Message::propertyExists (const std::string & *name*) const [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.25 virtual void cms::Message::setBooleanProperty (const std::string & *name,* bool *value*) [pure virtual]

Sets a boolean property.

Parameters

name	The name of the property to retrieve.
value	The value for the named property.

Exceptions

CMSException *(p. 640)*	- if the name is an empty string.
MessageNotWriteable-Exception *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.26 virtual void cms::Message::setByteProperty (const std::string & *name,* unsigned char *value*) [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. 640)*	- if the name is an empty string.
MessageNotWriteable-Exception *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.27 virtual void cms::Message::setCMSCorrelationID (const std::string & *correlationId*) [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. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.28 virtual void cms::Message::setCMSDeliveryMode (int *mode*) [pure virtual]

Sets the **DeliveryMode** (p. 925) 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. 925) enumerated value.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.29 virtual void cms::Message::setCMSDestination (const Destination * *destination*) [pure virtual]

Sets the **Destination** (p. 936) 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. 936) Object

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.30 virtual void cms::Message::setCMSExpiration (long long *expireTime*) [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.31 virtual void cms::Message::setCMSMessageID (const std::string & *id*) [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

CMSException *(p. 640)*	- if an internal error occurs.

6.341.3.32 virtual void cms::Message::setCMSPriority (int *priority*) [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.33 virtual void cms::Message::setCMSRedelivered (bool *redelivered*) [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.34 virtual void cms::Message::setCMSReplyTo (const cms::Destination * *destination*) [pure virtual]

Sets the **Destination** (p. 936) 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. 1722) object or a **Topic** (p. 2141) 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. 936) to which to send a response to this message

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.35 virtual void cms::Message::setCMSTimestamp (long long *timeStamp*) [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. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.36 virtual void cms::Message::setCMSType (const std::string & *type*) [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. 1435)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.37 virtual void cms::Message::setDoubleProperty (const std::string & *name*, double *value*) [pure virtual]

Sets a double property.

Parameters

name	The name of the property to retrieve.
value	The value for the named property.

Exceptions

CMSException *(p. 640)*	- if the name is an empty string.
MessageNotWriteableException *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.38 virtual void cms::Message::setFloatProperty (const std::string & *name*, float *value*) [pure virtual]

Sets a float property.

Parameters

name	The name of the property to retrieve.
value	The value for the named property.

Exceptions

CMSException *(p. 640)*	- if the name is an empty string.
MessageNotWriteable-Exception *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.39 virtual void cms::Message::setIntProperty (const std::string & *name,* int *value*) [pure virtual]

Sets a int property.

Parameters

name	The name of the property to retrieve.
value	The value for the named property.

Exceptions

CMSException *(p. 640)*	- if the name is an empty string.
MessageNotWriteable-Exception *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 232), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 232), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 232).

6.341.3.40 virtual void cms::Message::setLongProperty (const std::string & *name,* long long *value*) [pure virtual]

Sets a long property.

Parameters

name	The name of the property to retrieve.
value	The value for the named property.

Exceptions

CMSException *(p. 640)*	- if the name is an empty string.
MessageNotWriteable-Exception *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 233), **activemq::commands::ActiveMQMessage-**

Template< cms::StreamMessage > (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 233), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 233).

6.341.3.41 virtual void cms::Message::setShortProperty (const std::string & *name,* short *value*) [pure virtual]

Sets a short property.

Parameters

name	The name of the property to retrieve.
value	The value for the named property.

Exceptions

CMSException *(p. 640)*	- if the name is an empty string.
MessageNotWriteable-Exception *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 233), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 233).

6.341.3.42 virtual void cms::Message::setStringProperty (const std::string & *name,* const std::string & *value*) [pure virtual]

Sets a string property.

Parameters

name	The name of the property to retrieve.
value	The value for the named property.

Exceptions

CMSException *(p. 640)*	- if the name is an empty string.
MessageNotWriteable-Exception *(p. 1490)*	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 233), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 233), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 233).

6.341.4 Field Documentation

6.341.4.1 const int cms::Message::DEFAULT_DELIVERY_MODE [static]

The Default delivery mode for **Message** (p. 1426) Producers is PERSISTENT.

6.341.4.2 const int cms::Message::DEFAULT_MSG_PRIORITY [static]

The Default priority assigned to a **Message** (p. 1426) is 4.

6.341.4.3 const long long cms::Message::DEFAULT_TIME_TO_LIVE [static]

The Default Time to Live for a **Message** (p. 1426) Producer is unlimited, the message will never expire.

The documentation for this class was generated from the following file:

- src/main/cms/**Message.h**

6.342 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **MessageAck** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ConsumerId** > **consumerId**
- unsigned char **ackType**
- **Pointer**< **MessageId** > **firstMessageId**
- **Pointer**< **MessageId** > **lastMessageId**
- int **messageCount**

6.342.1 Constructor & Destructor Documentation

6.342.1.1 activemq::commands::MessageAck::MessageAck ( )

6.342.1.2 virtual activemq::commands::MessageAck::~MessageAck ( ) [virtual]

6.342.2 Member Function Documentation

6.342.2.1 virtual MessageAck* activemq::commands::MessageAck::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.342.2.2 virtual void activemq::commands::MessageAck::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. 382).

6.342.2.3 virtual bool activemq::commands::MessageAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.342.2.4 virtual unsigned char activemq::commands::MessageAck::getAckType ( ) const [virtual]

6.342.2.5 virtual const Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ( ) const [virtual]

6.342.2.6 virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ( ) [virtual]

6.342.2.7 virtual unsigned char activemq::commands::MessageAck::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.342.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination ( ) const [virtual]

6.342.2.9 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination ( ) [virtual]

6.342.2.10 virtual const Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ( ) const [virtual]

6.342.2.11 virtual Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ( ) [virtual]

6.342.2.12 virtual const Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ( ) const [virtual]

6.342.2.13 virtual Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ( ) [virtual]

6.342.2.14 virtual int activemq::commands::MessageAck::getMessageCount ( ) const [virtual]

6.342.2.15 virtual const Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId ( ) const [virtual]

6.342.2.16 virtual Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId ( ) [virtual]

6.342.2.17 virtual bool activemq::commands::MessageAck::isMessageAck ( ) const [inline, virtual]

Returns

an answer of true to the **isMessageAck()** (p. 1451) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.342.2.18 virtual void activemq::commands::MessageAck::setAckType (unsigned char *ackType*) [virtual]

6.342.2.19 virtual void activemq::commands::MessageAck::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.342.2.20 virtual void activemq::commands::MessageAck::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.342.2.21 virtual void activemq::commands::MessageAck::setFirstMessageId (const Pointer< MessageId > & *firstMessageId*) [virtual]

6.342.2.22 virtual void activemq::commands::MessageAck::setLastMessageId (const Pointer< MessageId > & *lastMessageId*) [virtual]

6.342.2.23 virtual void activemq::commands::MessageAck::setMessageCount (int *messageCount*) [virtual]

6.342.2.24 virtual void activemq::commands::MessageAck::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.342.2.25 virtual std::string activemq::commands::MessageAck::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.342.2.26 virtual Pointer<Command> activemq::commands::MessageAck::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.342.3 Field Documentation

6.342.3.1 unsigned char activemq::commands::MessageAck::ackType [protected]

6.342.3.2 Pointer<ConsumerId> activemq::commands::MessageAck::consumerId [protected]

6.342.3.3 Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination [protected]

6.342.3.4 Pointer<MessageId> activemq::commands::MessageAck::firstMessageId [protected]

6.342.3.5 const unsigned char activemq::commands::MessageAck::ID_MESSAGEACK = 22 [static]

6.342.3.6 Pointer<MessageId> activemq::commands::MessageAck::lastMessageId [protected]

6.342.3.7 int activemq::commands::MessageAck::messageCount [protected]

6.342.3.8 Pointer<TransactionId> activemq::commands::MessageAck::transactionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageAck.h**

6.343 activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1452).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/MessageAckMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual ~**MessageAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.343.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1452).

NOTE!: This file is 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::generated::MessageAckMarshaller::MessageAckMarshaller ( ) [inline]

6.343.2.2 virtual activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::∼MessageAckMarshaller ( ) [inline, virtual]

6.343.3 Member Function Documentation

6.343.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.343.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.343.3.3 virtual void activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::looseMarshal (OpenWireFormat ∗ *format,* commands::DataStructure ∗ *command,* decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.343.3.4 virtual void activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.343.3.5 virtual int activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.343.3.6 virtual void activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.343.3.7 virtual void activemq::wireformat::openwire::marshal::generated::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageAckMarshaller.h**

6.344 cms::MessageConsumer Class Reference

A client uses a **MessageConsumer** (p. 1455) to received messages from a destination.

`#include <src/main/cms/MessageConsumer.h>`

Inheritance diagram for cms::MessageConsumer:

Public Member Functions

- virtual ~**MessageConsumer** () throw ()
- virtual **Message** * **receive** ()=0

  *Synchronously Receive a **Message** (p. 1426).*
- virtual **Message** * **receive** (int millisecs)=0

  *Synchronously Receive a **Message** (p. 1426), time out after defined interval.*
- virtual **Message** * **receiveNoWait** ()=0

  *Receive a **Message** (p. 1426), does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (**MessageListener** *listener)=0

  *Sets the **MessageListener** (p. 1485) that this class will send notifs on.*
- virtual **MessageListener** * **getMessageListener** () **const** =0

  *Gets the **MessageListener** (p. 1485) that this class will send mew **Message** (p. 1426) notification events to.*
- virtual std::string **getMessageSelector** () **const** =0

  Gets this message consumer's message selector expression.

6.344.1 Detailed Description

A client uses a **MessageConsumer** (p. 1455) 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. 1485) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the **MessageListener** (p. 1485)'s `onMessage` method.

When the **MessageConsumer** (p. 1455)'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.

While the **MessageConsumer** (p. 1455) implements the **Startable** (p. 1963) and **Stoppable** (p. 2016) interfaces it is not required to implement these methods and can throw an UnsupportedOperation exception if they are not available for the given CMS provider.

See also

MessageListener (p. 1485)

Since

1.0

6.344.2 Constructor & Destructor Documentation

6.344.2.1 virtual cms::MessageConsumer::~MessageConsumer ( ) throw () [virtual]

6.344.3 Member Function Documentation

6.344.3.1 virtual MessageListener* cms::MessageConsumer::getMessageListener ( ) const [pure virtual]

Gets the **MessageListener** (p. 1485) that this class will send mew **Message** (p. 1426) notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQConsumer** (p. 186), and **activemq::cmsutil::CachedConsumer** (p. 570).

6.344.3.2 virtual std::string cms::MessageConsumer::getMessageSelector ( ) const [pure virtual]

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQConsumer** (p. 186), and **activemq::cmsutil::CachedConsumer** (p. 571).

6.344.3.3 virtual Message* cms::MessageConsumer::receive ( ) [pure virtual]

Synchronously Receive a **Message** (p. 1426).

Returns

new message which the caller owns and must delete.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQConsumer** (p. 187), and **activemq::cmsutil::CachedConsumer** (p. 571).

6.344.3.4 virtual Message* cms::MessageConsumer::receive (int *millisecs*) [pure virtual]

Synchronously Receive a **Message** (p. 1426), time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQConsumer** (p. 187), and **activemq::cmsutil::CachedConsumer** (p. 571).

6.344.3.5 virtual Message* cms::MessageConsumer::receiveNoWait ( ) [pure virtual]

Receive a **Message** (p. 1426), 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

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQConsumer** (p. 188), and **activemq::cmsutil::CachedConsumer** (p. 571).

6.344.3.6 virtual void cms::MessageConsumer::setMessageListener (MessageListener * *listener*) [pure virtual]

Sets the **MessageListener** (p. 1485) that this class will send notifs on.

Parameters

listener	The listener of messages received by this consumer.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQConsumer** (p. 188), and **activemq::cmsutil::CachedConsumer** (p. 572).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageConsumer.h**

6.345 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the **CmsTemplate** (p. 649).

`#include <src/main/activemq/cmsutil/MessageCreator.h>`

Public Member Functions

- virtual ~**MessageCreator** () throw ()
- virtual **cms::Message** * **createMessage** (**cms::Session** *session)=0

  Creates a message from the given session.

6.345.1 Detailed Description

Creates the user-defined message to be sent by the **CmsTemplate** (p. 649).

6.345.2 Constructor & Destructor Documentation

6.345.2.1 virtual activemq::cmsutil::MessageCreator::~MessageCreator ( ) throw () [inline, virtual]

6.345.3 Member Function Documentation

6.345.3.1 virtual cms::Message* activemq::cmsutil::MessageCreator::createMessage (cms::Session * *session*) [pure virtual]

Creates a message from the given session.

Parameters

session	the CMS Session

Exceptions

cms::CMSException *(p. 640)*	if thrown by CMS API methods

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**MessageCreator.h**

6.346 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **MessageDispatch** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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_MESSAGEDISPATCH** = 21

Protected Attributes

- **Pointer< ConsumerId > consumerId**
- **Pointer< ActiveMQDestination > destination**
- **Pointer< Message > message**
- int **redeliveryCounter**

6.346.1 Constructor & Destructor Documentation

6.346.1.1 activemq::commands::MessageDispatch::MessageDispatch ( )

6.346.1.2 virtual activemq::commands::MessageDispatch::~MessageDispatch ( ) [virtual]

6.346.2 Member Function Documentation

6.346.2.1 virtual MessageDispatch* activemq::commands::MessageDispatch::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.346.2.2 virtual void activemq::commands::MessageDispatch::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. 382).

6.346.2.3 virtual bool activemq::commands::MessageDispatch::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.346.2.4 virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ( ) const [virtual]

6.346.2.5 virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ( ) [virtual]

6.346.2.6 virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.346.2.7 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ( ) const [virtual]

6.346.2.8 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ( ) [virtual]

6.346.2.9 virtual const Pointer<Message>& activemq::commands::MessageDispatch::getMessage ( ) const [virtual]

6.346.2.10 virtual Pointer<Message>& activemq::commands::MessageDispatch::getMessage ( ) [virtual]

6.346.2.11 virtual int activemq::commands::MessageDispatch::getRedeliveryCounter ( ) const [virtual]

6.346.2.12 virtual bool activemq::commands::MessageDispatch::isMessageDispatch ( ) const [inline, virtual]

Returns

an answer of true to the **isMessageDispatch()** (p. 1461) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.346.2.13 virtual void activemq::commands::MessageDispatch::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.346.2.14 virtual void activemq::commands::MessageDispatch::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.346.2.15 virtual void activemq::commands::MessageDispatch::setMessage (const Pointer< Message > & *message*) [virtual]

6.346.2.16 virtual void activemq::commands::MessageDispatch::setRedeliveryCounter (int *redeliveryCounter*) [virtual]

6.346.2.17 virtual std::string activemq::commands::MessageDispatch::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.346.2.18 virtual Pointer<Command> activemq::commands::MessageDispatch::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.346.3 Field Documentation

6.346.3.1 Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId [protected]

6.346.3.2 Pointer<ActiveMQDestination> activemq::commands::MessageDispatch::destination [protected]

6.346.3.3 const unsigned char activemq::commands::MessageDispatch::ID_MESSAGEDISPATCH = 21 [static]

6.346.3.4 Pointer<Message> activemq::commands::MessageDispatch::message [protected]

6.346.3.5 int activemq::commands::MessageDispatch::redeliveryCounter [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatch.h**

6.347 activemq::core::MessageDispatchChannel Class Reference

`#include <src/main/activemq/core/MessageDispatchChannel.h>`

Inheritance diagram for activemq::core::MessageDispatchChannel:

Public Member Functions

- virtual ~**MessageDispatchChannel** ()
- virtual void **enqueue** (**const Pointer**< **MessageDispatch** > &message)=0

  Add a Message to the Channel behind all pending message.
- virtual void **enqueueFirst** (**const Pointer**< **MessageDispatch** > &message)=0

  Add a message to the front of the Channel.
- virtual bool **isEmpty** () **const** =0
- virtual bool **isClosed** () **const** =0
- virtual bool **isRunning** () **const** =0
- virtual **Pointer**< **MessageDispatch** > **dequeue** (long long timeout)=0

  Used to get an enqueued message.
- virtual **Pointer**< **MessageDispatch** > **dequeueNoWait** ()=0

  Used to get an enqueued message if there is one queued right now.

- virtual **Pointer**< **MessageDispatch** > **peek** () **const** =0

  Peek in the Queue and return the first message in the Channel without removing it from the channel.
- virtual void **start** ()=0

  Starts dispatch of messages from the Channel.
- virtual void **stop** ()=0

  Stops dispatch of message from the Channel.
- virtual void **close** ()=0

  Close this channel no messages will be dispatched after this method is called.
- virtual void **clear** ()=0

  Clear the Channel, all pending messages are removed.
- virtual int **size** () **const** =0
- virtual std::vector< **Pointer** < **MessageDispatch** > > **removeAll** ()=0

  Remove all messages that are currently in the Channel and return them as a list of Messages.

6.347.1 Constructor & Destructor Documentation

6.347.1.1 virtual activemq::core::MessageDispatchChannel::~MessageDispatchChannel ( ) [inline, virtual]

6.347.2 Member Function Documentation

6.347.2.1 virtual void activemq::core::MessageDispatchChannel::clear ( ) [pure virtual]

Clear the Channel, all pending messages are removed.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1895), and **activemq::core::FifoMessageDispatchChannel** (p. 1024).

6.347.2.2 virtual void activemq::core::MessageDispatchChannel::close ( ) [pure virtual]

Close this channel no messages will be dispatched after this method is called.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1895), and **activemq::core::FifoMessageDispatchChannel** (p. 1024).

6.347.2.3 virtual Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue (long long *timeout*) [pure virtual]

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 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 consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

ActiveMQException	

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1895), and **activemq::core::FifoMessageDispatchChannel** (p. 1024).

6.347.2.4 virtual Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait ( ) [pure virtual]

Used to get an enqueued message if there is one queued right now.

If there is no waiting message than this method returns Null.

Returns

a message if there is one in the queue.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1895), and **activemq::core::FifoMessageDispatchChannel** (p. 1025).

6.347.2.5 virtual void activemq::core::MessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & *message*) [pure virtual]

Add a Message to the Channel behind all pending message.

Parameters

message	- The message to add to the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1895), and **activemq::core::FifoMessageDispatchChannel** (p. 1025).

6.347.2.6 virtual void activemq::core::MessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & *message*) [pure virtual]

Add a message to the front of the Channel.

Parameters

message	- The Message to add to the front of the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1896), and **activemq::core::FifoMessageDispatchChannel** (p. 1025).

6.347.2.7 virtual bool activemq::core::MessageDispatchChannel::isClosed ( ) const [pure virtual]

Returns

has the Queue been closed.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1896), and **activemq::core::FifoMessageDispatchChannel** (p. 1025).

6.347.2.8 virtual bool activemq::core::MessageDispatchChannel::isEmpty ( ) const [pure virtual]

Returns

true if there are no messages in the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1896), and **activemq::core::FifoMessageDispatchChannel** (p. 1025).

6.347.2.9 virtual bool activemq::core::MessageDispatchChannel::isRunning ( ) const [pure virtual]

Returns

true if the Channel currently running and will dequeue message.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1896), and **activemq::core::FifoMessageDispatchChannel** (p. 1026).

6.347.2.10 virtual Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek ( ) const [pure virtual]

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.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1897), and **activemq::core::FifoMessageDispatchChannel** (p. 1027).

6.347.2.11 virtual std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ( ) [pure virtual]

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.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1897), and **activemq::core::FifoMessageDispatchChannel** (p. 1027).

6.347.2.12 virtual int activemq::core::MessageDispatchChannel::size ( ) const [pure virtual]

Returns

the number of Messages currently in the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1897), and **activemq::core::FifoMessageDispatchChannel** (p. 1027).

6.347.2.13 virtual void activemq::core::MessageDispatchChannel::start ( ) [pure virtual]

Starts dispatch of messages from the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1898), and **activemq::core::FifoMessageDispatchChannel** (p. 1027).

6.347.2.14 virtual void activemq::core::MessageDispatchChannel::stop ( ) [pure virtual]

Stops dispatch of message from the Channel.

Implemented in **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1898), and **activemq::core::FifoMessageDispatchChannel** (p. 1027).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**MessageDispatchChannel.h**

6.348 activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1466).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/MessageDispatchMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual ~**MessageDispatchMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.348.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1466).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.348.2 Constructor & Destructor Documentation

6.348.2.1 activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]

6.348.2.2 virtual activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::~MessageDispatchMarshaller ( ) [inline, virtual]

6.348.3 Member Function Documentation

6.348.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.348.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.348.3.3 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.348.3.4 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.348.3.5 virtual int activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.348.3.6 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.348.3.7 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageDispatchMarshaller.h**

6.349 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **MessageDispatchNotification** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **deliverySequenceId**
- **Pointer**< **MessageId** > **messageId**

6.349.1 Constructor & Destructor Documentation

6.349.1.1 activemq::commands::MessageDispatchNotification::MessageDispatchNotification ( )

6.349.1.2 virtual activemq::commands::MessageDispatchNotification::~MessageDispatchNotification ( ) [virtual]

6.349.2 Member Function Documentation

6.349.2.1 virtual MessageDispatchNotification* activemq::commands::MessageDispatchNotification::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.349.2.2 virtual void activemq::commands::MessageDispatchNotification::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. 382).

6.349.2.3 virtual bool activemq::commands::MessageDispatchNotification::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.349.2.4 virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ( ) const [virtual]

6.349.2.5 virtual Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ( ) [virtual]

6.349.2.6 virtual unsigned char activemq::commands::MessageDispatchNotification::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.349.2.7 virtual long long activemq::commands::MessageDispatchNotification::getDeliverySequenceId ( ) const [virtual]

6.349.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ( ) const [virtual]

6.349.2.9 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ( ) [virtual]

6.349.2.10 virtual const Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId ( ) const [virtual]

6.349.2.11 virtual Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId ( ) [virtual]

6.349.2.12 virtual bool activemq::commands::MessageDispatchNotification::isMessageDispatchNotification ( ) const [inline, virtual]

Returns

an answer of true to the **isMessageDispatchNotification()** (p. 1471) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.349.2.13 virtual void activemq::commands::MessageDispatchNotification::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.349.2.14 virtual void activemq::commands::MessageDispatchNotification::setDeliverySequenceId (long long *deliverySequenceId*) [virtual]

6.349.2.15 virtual void activemq::commands::MessageDispatchNotification::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.349.2.16 virtual void activemq::commands::MessageDispatchNotification::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]

6.349.2.17 virtual std::string activemq::commands::MessageDispatchNotification::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.349.2.18 virtual Pointer<Command> activemq::commands::MessageDispatchNotification::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.349.3 Field Documentation

6.349.3.1 Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId [protected]

6.349.3.2 long long activemq::commands::MessageDispatchNotification::deliverySequenceId [protected]

6.349.3.3 Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination [protected]

6.349.3.4 const unsigned char activemq::commands::MessageDispatchNotification::ID_MESSAGEDISPATCHNOTIFICATION = 90 [static]

6.349.3.5 Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatchNotification.h**

6.350 activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1472).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/MessageDispatchNotificationMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::MessageDispatchNotification-Marshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ∼**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.350.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1472).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.350.2 Constructor & Destructor Documentation

6.350.2.1 activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::-MessageDispatchNotificationMarshaller ( ) [inline]

6.350.2.2 virtual activemq::wireformat::openwire::marshal::generated::MessageDispatch-NotificationMarshaller::∼MessageDispatchNotificationMarshaller ( ) [inline, virtual]

6.350.3 Member Function Documentation

6.350.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal-::generated::MessageDispatchNotificationMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.350.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.350.3.3 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.350.3.4 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.350.3.5 virtual int activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.350.3.6 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.350.3.7 virtual void activemq::wireformat::openwire::marshal::generated::MessageDispatchNotificationMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageDispatchNotificationMarshaller.h**

6.351 cms::MessageEnumeration Class Reference

Defines an object that enumerates a collection of Messages.

`#include <src/main/cms/MessageEnumeration.h>`

Inheritance diagram for cms::MessageEnumeration:

Public Member Functions

- virtual ~**MessageEnumeration** () throw ()
- virtual bool **hasMoreMessages** ()=0

  *Returns true if there are more **Message** (p. 1426) in the Browser that can be retrieved via the `nextMessage` method.*
- virtual **cms::Message** * **nextMessage** ()=0

  *Returns the Next **Message** (p. 1426) in the **Queue** (p. 1722) if one is present, if no more **Message** (p. 1426)'s are available then an Exception is thrown.*

6.351.1 Detailed Description

Defines an object that enumerates a collection of Messages.

The client calls the hasMoreMessages method to determine if a **Message** (p. 1426) is available. If a **Message** (p. 1426) is available the client calls the nextMessage method to retrieve that **Message** (p. 1426), calling nextMessage when a **Message** (p. 1426) is not available results in an exception.

Since

2.1

6.351.2 Constructor & Destructor Documentation

6.351.2.1 virtual cms::MessageEnumeration::~MessageEnumeration ( ) throw () [virtual]

6.351.3 Member Function Documentation

6.351.3.1 virtual bool cms::MessageEnumeration::hasMoreMessages ( ) [pure virtual]

Returns true if there are more **Message** (p. 1426) in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

Returns

true if more **Message** (p. 1426)'s are available in the Browser.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 254).

6.351.3.2 virtual cms::Message* cms::MessageEnumeration::nextMessage ( ) [pure virtual]

Returns the Next **Message** (p. 1426) in the **Queue** (p. 1722) if one is present, if no more **Message** (p. 1426)'s are available then an Exception is thrown.

If a **Message** (p. 1426) object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

Returns

The next **Message** (p. 1426) in the **Queue** (p. 1722).

Exceptions

CMSException *(p. 640)*	if no more **Message** (p. 1426)'s currently in the **Queue** (p. 1722).

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 254).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageEnumeration.h**

6.352 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2020) or **BytesMessage** (p. 557) is being read.

`#include <src/main/cms/MessageEOFException.h>`

Inheritance diagram for cms::MessageEOFException:

Public Member Functions

- **MessageEOFException** ()
- **MessageEOFException** (**const MessageEOFException** &ex)
- **MessageEOFException** (**const** std::string &message)
- **MessageEOFException** (**const** std::string &message, **const** std::exception *cause)
- **MessageEOFException** (**const** std::string &message, **const** std::exception *cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**MessageEOFException** () throw ()

6.352.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2020) or **BytesMessage** (p. 557) is being read.

Since

1.3

6.352.2 Constructor & Destructor Documentation

6.352.2.1 cms::MessageEOFException::MessageEOFException ( )

6.352.2.2 cms::MessageEOFException::MessageEOFException (const MessageEOFException & *ex*)

6.352.2.3 cms::MessageEOFException::MessageEOFException (const std::string & *message*)

6.352.2.4 cms::MessageEOFException::MessageEOFException (const std::string & *message,* const std::exception * *cause*)

6.352.2.5 cms::MessageEOFException::MessageEOFException (const std::string & *message,* const std::exception * *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.352.2.6 virtual cms::MessageEOFException::~MessageEOFException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageEOFException.h**

6.353 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** ()
- **MessageFormatException** (**const MessageFormatException** &ex)
- **MessageFormatException** (**const** std::string &message)
- **MessageFormatException** (**const** std::string &message, **const** std::exception *cause)
- **MessageFormatException** (**const** std::string &message, **const** std::exception *cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**MessageFormatException** () throw ()

6.353.1 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. 2026) is used to read a boolean value.

Since

1.3

6.353.2 Constructor & Destructor Documentation

6.353.2.1 cms::MessageFormatException::MessageFormatException ( )

6.353.2.2 cms::MessageFormatException::MessageFormatException (const MessageFormatException & *ex*)

6.353.2.3 **cms::MessageFormatException::MessageFormatException (const std::string & *message*)**

6.353.2.4 **cms::MessageFormatException::MessageFormatException (const std::string & *message,* const std::exception * *cause*)**

6.353.2.5 **cms::MessageFormatException::MessageFormatException (const std::string & *message,* const std::exception * *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)**

6.353.2.6 **virtual cms::MessageFormatException::~MessageFormatException ( ) throw ()** `[virtual]`

The documentation for this class was generated from the following file:

- src/main/cms/**MessageFormatException.h**

6.354 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)
- **MessageId** (**const** std::string &messageKey)
- **MessageId** (**const Pointer**< **ProducerInfo** > &producerInfo, long long **producerSequenceId**)
- **MessageId** (**const Pointer**< **ProducerId** > &**producerId**, long long **producerSequenceId**)
- **MessageId** (**const** std::string &**producerId**, long long **producerSequenceId**)
- virtual ~**MessageId** ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **MessageId** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- void **setValue** (**const** std::string &key)
- void **setTextView** (**const** std::string &key)
- 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**

6.354.1 Member Typedef Documentation

6.354.1.1 typedef decaf::lang::PointerComparator<MessageId> activemq::commands::MessageId::COMPARATOR

6.354.2 Constructor & Destructor Documentation

6.354.2.1 activemq::commands::MessageId::MessageId ( )

6.354.2.2 activemq::commands::MessageId::MessageId (const MessageId & *other*)

6.354.2.3 activemq::commands::MessageId::MessageId (const std::string & *messageKey*)

6.354.2.4 activemq::commands::MessageId::MessageId (const Pointer< ProducerInfo > & *producerInfo,* long long *producerSequenceId*)

6.354.2.5 activemq::commands::MessageId::MessageId (const Pointer< ProducerId > & *producerId,* long long *producerSequenceId*)

6.354.2.6 activemq::commands::MessageId::MessageId (const std::string & *producerId,* long long *producerSequenceId*)

6.354.2.7 virtual activemq::commands::MessageId::~MessageId ( ) [virtual]

6.354.3 Member Function Documentation

6.354.3.1 virtual MessageId* activemq::commands::MessageId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.354.3.2 virtual int activemq::commands::MessageId::compareTo (const MessageId & *value*) const [virtual]

6.354.3.3 virtual void activemq::commands::MessageId::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. 879).

6.354.3.4 virtual bool activemq::commands::MessageId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.354.3.5 virtual bool activemq::commands::MessageId::equals (const MessageId & *value*) const [virtual]

6.354.3.6 virtual long long activemq::commands::MessageId::getBrokerSequenceId ( ) const [virtual]

6.354.3.7 virtual unsigned char activemq::commands::MessageId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.354.3.8 virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ( ) const [virtual]

6.354.3.9 virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ( ) [virtual]

6.354.3.10 virtual long long activemq::commands::MessageId::getProducerSequenceId ( ) const [virtual]

6.354.3.11 virtual bool activemq::commands::MessageId::operator< (const MessageId & *value*) const [virtual]

6.354.3.12 MessageId& activemq::commands::MessageId::operator= (const MessageId & *other*)

6.354.3.13 virtual bool activemq::commands::MessageId::operator== (const MessageId & *value*) const [virtual]

6.354.3.14 virtual void **activemq::commands::MessageId::setBrokerSequenceId** (long long *brokerSequenceId*) [virtual]

6.354.3.15 virtual void **activemq::commands::MessageId::setProducerId** (const Pointer< ProducerId > & *producerId*) [virtual]

6.354.3.16 virtual void **activemq::commands::MessageId::setProducerSequenceId** (long long *producerSequenceId*) [virtual]

6.354.3.17 void **activemq::commands::MessageId::setTextView** (const std::string & *key*)

6.354.3.18 void **activemq::commands::MessageId::setValue** (const std::string & *key*)

6.354.3.19 virtual std::string **activemq::commands::MessageId::toString** ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.354.4 Field Documentation

6.354.4.1 long long **activemq::commands::MessageId::brokerSequenceId** [protected]

6.354.4.2 const unsigned char **activemq::commands::MessageId::ID_MESSAGEID** = 110 [static]

6.354.4.3 Pointer<ProducerId> **activemq::commands::MessageId::producerId** [protected]

6.354.4.4 long long **activemq::commands::MessageId::producerSequenceId** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageId.h**

6.355 activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1482).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/MessageIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual ~**MessageIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.

- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.355.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1482).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.355.2 Constructor & Destructor Documentation

6.355.2.1 activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::MessageId-Marshaller ( ) [inline]

6.355.2.2 virtual activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::∼MessageId-Marshaller ( ) [inline, virtual]

6.355.3 Member Function Documentation

6.355.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::-MessageIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.355.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller-::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.355.3.3 virtual void activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.355.3.4 virtual void activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.355.3.5 virtual int activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.355.3.6 virtual void activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.355.3.7 virtual void activemq::wireformat::openwire::marshal::generated::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageIdMarshaller.h**

6.356 cms::MessageListener Class Reference

A **MessageListener** (p. 1485) object is used to receive asynchronously delivered messages.

`#include <src/main/cms/MessageListener.h>`

Public Member Functions

- virtual ~**MessageListener** () throw ()
- virtual void **onMessage** (**const Message** *message)=0 throw ()

  *Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 1426) types.*

6.356.1 Detailed Description

A **MessageListener** (p. 1485) object is used to receive asynchronously delivered messages.

Since

1.0

6.356.2 Constructor & Destructor Documentation

6.356.2.1 virtual cms::MessageListener::~MessageListener ( ) throw () [virtual]

6.356.3 Member Function Documentation

6.356.3.1 virtual void cms::MessageListener::onMessage (const Message * *message*) throw () [pure virtual]

Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 1426) 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. 1426).

It is considered a programming error for this method to throw an exception. The method has been tagged with the 'throw()' qualifier, this implies that you application will segfault if you throw an error from an implementation of this method.

Parameters

message	**Message** (p. 1426) object {const} pointer recipient does not own.

The documentation for this class was generated from the following file:

- src/main/cms/**MessageListener.h**

6.357 activemq::wireformat::openwire::marshal::generated::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1486).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/MessageMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual ~**MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.357.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1486).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.357.2 Constructor & Destructor Documentation

6.357.2.1 activemq::wireformat::openwire::marshal::generated::MessageMarshaller::MessageMarshaller ( ) [inline]

6.357.2.2 virtual activemq::wireformat::openwire::marshal::generated::MessageMarshaller::∼MessageMarshaller ( ) [inline, virtual]

6.357.3 Member Function Documentation

6.357.3.1 virtual void activemq::wireformat::openwire::marshal::generated::MessageMarshaller::looseMarshal (OpenWireFormat ∗ *format,* commands::DataStructure ∗ *command,* decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 221), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319).

6.357.3.2 virtual void activemq::wireformat::openwire::marshal::generated::MessageMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 127), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 319).

6.357.3.3 virtual int activemq::wireformat::openwire::marshal::generated::MessageMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 144), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 292), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320).

6.357.3.4 virtual void activemq::wireformat::openwire::marshal::generated::MessageMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 145), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 293), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320).

6.357.3.5 virtual void activemq::wireformat::openwire::marshal::generated::MessageMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::ActiveMQBlobMessageMarshaller** (p. 128), **activemq::wireformat::openwire::marshal::generated::ActiveMQBytesMessageMarshaller** (p. 145), **activemq::wireformat::openwire::marshal::generated::ActiveMQMapMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::generated::ActiveMQMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::generated::ActiveMQObjectMessageMarshaller** (p. 238), **activemq::wireformat::openwire::marshal::generated::ActiveMQStreamMessageMarshaller** (p. 293), and **activemq::wireformat::openwire::marshal::generated::ActiveMQTextMessageMarshaller** (p. 320).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessageMarshaller.h**

6.358 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** ()
- **MessageNotReadableException** (**const MessageNotReadableException** &ex)
- **MessageNotReadableException** (**const** std::string &message)
- **MessageNotReadableException** (**const** std::string &message, **const** std::exception ∗cause)
- **MessageNotReadableException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ∼**MessageNotReadableException** () throw ()

6.358.1 Detailed Description

This exception must be thrown when a CMS client attempts to read a write-only message.

Since

1.3

6.358.2 Constructor & Destructor Documentation

6.358.2.1 cms::MessageNotReadableException::MessageNotReadableException ( )

6.358.2.2 cms::MessageNotReadableException::MessageNotReadableException (const MessageNotReadableException & *ex*)

6.358.2.3 cms::MessageNotReadableException::MessageNotReadableException (const std::string & *message*)

6.358.2.4 cms::MessageNotReadableException::MessageNotReadableException (const std::string & *message,* const std::exception ∗ *cause*)

6.358.2.5 cms::MessageNotReadableException::MessageNotReadableException (const std::string & *message,* const std::exception ∗ *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.358.2.6 virtual cms::MessageNotReadableException::∼MessageNotReadableException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageNotReadableException.h**

6.359 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** ()
- **MessageNotWriteableException** (**const MessageNotWriteableException** &ex)
- **MessageNotWriteableException** (**const** std::string &message)
- **MessageNotWriteableException** (**const** std::string &message, **const** std::exception ∗cause)
- **MessageNotWriteableException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ∼**MessageNotWriteableException** () throw ()

6.359.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since

1.3

6.359.2 Constructor & Destructor Documentation

6.359.2.1 cms::MessageNotWriteableException::MessageNotWriteableException ( )

6.359.2.2 cms::MessageNotWriteableException::MessageNotWriteableException (const MessageNotWriteableException & *ex*)

6.359.2.3 cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & *message*)

6.359.2.4 cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & *message,* const std::exception ∗ *cause*)

6.359.2.5 cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & *message,* const std::exception ∗ *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.359.2.6 virtual cms::MessageNotWriteableException::∼MessageNotWriteableException ( ) throw () `[virtual]`

The documentation for this class was generated from the following file:

- src/main/cms/**MessageNotWriteableException.h**

6.360 cms::MessageProducer Class Reference

A client uses a **`MessageProducer`** (p. 1491) object to send messages to a **Destination** (p. 936).

`#include <src/main/cms/MessageProducer.h>`

Inheritance diagram for cms::MessageProducer:

Public Member Functions

- virtual ~**MessageProducer** () throw ()
- virtual void **send** (**Message** *message)=0

  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

  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

  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

  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

  Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () **const** =0

  Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value)=0

  *Sets if **Message** (p. 1426) Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () **const** =0

  *Gets if **Message** (p. 1426) Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value)=0

  *Sets if **Message** (p. 1426) Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () **const** =0

  *Gets if **Message** (p. 1426) Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority)=0

  Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () **const** =0

  Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time)=0

  Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () **const** =0

  Gets the Time to Live that this producer sends messages with.

6.360.1 Detailed Description

A client uses a **MessageProducer** (p. 1491) object to send messages to a **Destination** (p. 936).

A **MessageProducer** (p. 1491) object is created by passing a **Destination** (p. 936) 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. 936) 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

6.360.2 Constructor & Destructor Documentation

6.360.2.1 virtual cms::MessageProducer::~MessageProducer ( ) throw () [virtual]

6.360.3 Member Function Documentation

6.360.3.1 virtual int cms::MessageProducer::getDeliveryMode ( ) const [pure virtual]

Gets the delivery mode for this Producer.

Returns

The **DeliveryMode** (p. 925)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 240), and **activemq::cmsutil::CachedProducer** (p. 574).

6.360.3.2 virtual bool cms::MessageProducer::getDisableMessageID ( ) const [pure virtual]

Gets if **Message** (p. 1426) Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 240), and **activemq::cmsutil::CachedProducer** (p. 574).

6.360.3.3 virtual bool cms::MessageProducer::getDisableMessageTimeStamp ( ) const [pure virtual]

Gets if **Message** (p. 1426) Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 240), and **activemq::cmsutil::CachedProducer** (p. 574).

6.360.3.4 virtual int cms::MessageProducer::getPriority ( ) const [pure virtual]

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 241), and **activemq::cmsutil::CachedProducer** (p. 575).

6.360.3.5 virtual long long cms::MessageProducer::getTimeToLive ( ) const [pure virtual]

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 241), and **activemq::cmsutil::CachedProducer** (p. 575).

6.360.3.6 virtual void cms::MessageProducer::send (Message * *message*) [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. 640)*	- if an internal error occurs while sending the message.
MessageFormatException *(p. 1478)*	- if an Invalid **Message** (p. 1426) is given.
InvalidDestination-Exception *(p. 1190)*	- if a client uses this method with a **MessageProducer** (p. 1491) with an invalid destination.
UnsupportedOperation-Exception *(p. 2190)*	- if a client uses this method with a **MessageProducer** (p. 1491) that did not specify a destination at creation time.

Implemented in **activemq::core::ActiveMQProducer** (p. 242), and **activemq::cmsutil::CachedProducer** (p. 575).

6.360.3.7 virtual void cms::MessageProducer::send (Message * *message,* int *deliveryMode,* int *priority,* long long *timeToLive*) [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

CMSException *(p. 640)*	- if an internal error occurs while sending the message.
MessageFormatException *(p. 1478)*	- if an Invalid **Message** (p. 1426) is given.
InvalidDestination-Exception *(p. 1190)*	- if a client uses this method with a **MessageProducer** (p. 1491) with an invalid destination.
UnsupportedOperation-Exception *(p. 2190)*	- if a client uses this method with a **MessageProducer** (p. 1491) that did not specify a destination at creation time.

Implemented in **activemq::core::ActiveMQProducer** (p. 242), and **activemq::cmsutil::CachedProducer** (p. 575).

6.360.3.8 virtual void cms::MessageProducer::send (const Destination * *destination,* Message * *message*) [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. 640)*	- if an internal error occurs while sending the message.
MessageFormatException *(p. 1478)*	- if an Invalid **Message** (p. 1426) is given.
InvalidDestination-Exception *(p. 1190)*	- if a client uses this method with a **MessageProducer** (p. 1491) with an invalid destination.
UnsupportedOperation-Exception *(p. 2190)*	- if a client uses this method with a **MessageProducer** (p. 1491) that did not specify a destination at creation time.

Implemented in **activemq::core::ActiveMQProducer** (p. 243), and **activemq::cmsutil::CachedProducer** (p. 576).

6.360.3.9 virtual void cms::MessageProducer::send (const Destination * *destination,* Message * *message,* int *deliveryMode,* int *priority,* long long *timeToLive*) [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

CMSException *(p. 640)*	- if an internal error occurs while sending the message.
MessageFormatException *(p. 1478)*	- if an Invalid **Message** (p. 1426) is given.
InvalidDestination-Exception *(p. 1190)*	- if a client uses this method with a **MessageProducer** (p. 1491) with an invalid destination.
UnsupportedOperation-Exception *(p. 2190)*	- if a client uses this method with a **MessageProducer** (p. 1491) that did not specify a destination at creation time.

Implemented in **activemq::core::ActiveMQProducer** (p. 243), and **activemq::cmsutil::CachedProducer** (p. 576).

6.360.3.10 virtual void cms::MessageProducer::setDeliveryMode (int *mode*) [pure virtual]

Sets the delivery mode for this Producer.

Parameters

mode	The **DeliveryMode** (p. 925)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 243), and **activemq::cmsutil::CachedProducer** (p. 577).

6.360.3.11 virtual void cms::MessageProducer::setDisableMessageID (bool *value*) [pure virtual]

Sets if **Message** (p. 1426) Ids are disabled for this Producer.

Parameters

value	boolean indicating enable / disable (true / false)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 244), and **activemq::cmsutil::CachedProducer** (p. 577).

6.360.3.12 virtual void cms::MessageProducer::setDisableMessageTimeStamp (bool *value*) [pure virtual]

Sets if **Message** (p. 1426) Time Stamps are disabled for this Producer.

Parameters

value	- boolean indicating enable / disable (true / false)

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 244), and **activemq::cmsutil::CachedProducer** (p. 577).

6.360.3.13 virtual void cms::MessageProducer::setPriority (int *priority*) [pure virtual]

Sets the Priority that this Producers sends messages at.

Parameters

priority	int value for Priority level

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 244), and **activemq::cmsutil::CachedProducer** (p. 578).

6.360.3.14 virtual void cms::MessageProducer::setTimeToLive (long long *time*) [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

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::core::ActiveMQProducer** (p. 244), and **activemq::cmsutil::CachedProducer** (p. 578).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageProducer.h**

6.361 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)

  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** () throw ()
- 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.

6.361.1 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

6.361.2 Constructor & Destructor Documentation

6.361.2.1 activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInterceptor (commands::Message * *message*, util::PrimitiveMap * *properties*)

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

6.361.2.2 virtual activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~MessagePropertyInterceptor ( ) throw () [virtual]

6.361.3 Member Function Documentation

6.361.3.1 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.

6.361.3.2 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.

6.361.3.3 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.

6.361.3.4 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.

6.361.3.5 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.

6.361.3.6 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.

6.361.3.7 virtual 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.

6.361.3.8 virtual std::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.

6.361.3.9 virtual 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.

6.361.3.10 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.

6.361.3.11 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.

6.361.3.12 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.

6.361.3.13 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.

6.361.3.14 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.

6.361.3.15 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.

6.361.3.16 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**

6.362 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **MessagePull** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **timeout**
- std::string **correlationId**
- **Pointer**< **MessageId** > **messageId**

6.362.1 Constructor & Destructor Documentation

6.362.1.1 activemq::commands::MessagePull::MessagePull ( )

6.362.1.2 virtual activemq::commands::MessagePull::~MessagePull ( ) [virtual]

6.362.2 Member Function Documentation

6.362.2.1 virtual MessagePull* activemq::commands::MessagePull::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.362.2.2 virtual void activemq::commands::MessagePull::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. 382).

6.362.2.3 virtual bool activemq::commands::MessagePull::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.362.2.4 virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ( ) const [virtual]

6.362.2.5 virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ( ) [virtual]

6.362.2.6 virtual const std::string& activemq::commands::MessagePull::getCorrelationId ( ) const [virtual]

6.362.2.7 virtual std::string& activemq::commands::MessagePull::getCorrelationId ( ) [virtual]

6.362.2.8 virtual unsigned char activemq::commands::MessagePull::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.362.2.9 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination ( ) const [virtual]

6.362.2.10 virtual Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination ( ) [virtual]

6.362.2.11 virtual const Pointer<MessageId>& activemq::commands::MessagePull::getMessageId ( ) const [virtual]

6.362.2.12 virtual Pointer<MessageId>& activemq::commands::MessagePull::getMessageId ( ) [virtual]

6.362.2.13 virtual long long activemq::commands::MessagePull::getTimeout ( ) const [virtual]

6.362.2.14 virtual void activemq::commands::MessagePull::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.362.2.15 virtual void activemq::commands::MessagePull::setCorrelationId (const std::string & *correlationId*) [virtual]

6.362.2.16 virtual void activemq::commands::MessagePull::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.362.2.17 virtual void activemq::commands::MessagePull::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]

6.362.2.18 virtual void activemq::commands::MessagePull::setTimeout (long long *timeout*) [virtual]

6.362.2.19 virtual std::string activemq::commands::MessagePull::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.362.2.20 virtual Pointer<Command> activemq::commands::MessagePull::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.362.3 Field Documentation

6.362.3.1 Pointer<ConsumerId> activemq::commands::MessagePull::consumerId [protected]

6.362.3.2 std::string activemq::commands::MessagePull::correlationId [protected]

6.362.3.3 Pointer<ActiveMQDestination> activemq::commands::MessagePull::destination [protected]

6.362.3.4 const unsigned char activemq::commands::MessagePull::ID_MESSAGEPULL = 20 [static]

6.362.3.5 Pointer<MessageId> activemq::commands::MessagePull::messageId [protected]

6.362.3.6 long long activemq::commands::MessagePull::timeout [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessagePull.h**

6.363 activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1506).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/MessagePullMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual ∼**MessagePullMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.363.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1506).

NOTE!: This file is 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::generated::MessagePullMarshaller::MessagePullMarshaller ( ) [inline]

6.363.2.2 virtual activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::∼MessagePullMarshaller ( ) [inline, virtual]

6.363.3 Member Function Documentation

6.363.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.363.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.363.3.3 virtual void activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.363.3.4 virtual void activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.363.3.5 virtual int activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.363.3.6 virtual void activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.363.3.7 virtual void activemq::wireformat::openwire::marshal::generated::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**MessagePullMarshaller.h**

6.364 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 1509) defines a base level **Transport** (p. 2161) class that is intended to be used in place of an a regular protocol **Transport** (p. 2161) suck 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** ()
- 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.
- void **setResponseBuilder** (**const Pointer**< **ResponseBuilder** > &responseBuilder)

  *Sets the **ResponseBuilder** (p. 1784) that this class uses to create Responses to Commands sent.*
- 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.
- **Pointer**< **wireformat::WireFormat** > **getWireFormat** () **const**

  Gets the currently set WireFormat.
- virtual void **oneway** (**const Pointer**< **Command** > &command)

  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command)

  Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command, unsigned int timeout)

  Sends the given command to the broker and then waits for the response.
- virtual void **setWireFormat** (**const Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP_UNUSED)
- virtual void **setTransportListener** (**TransportListener** *listener)

  Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () **const**

  Gets the observer of asynchronous events from this transport.
- virtual void **start** ()

  *Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.*
- virtual void **stop** ()

  *Stops the **Transport** (p. 2161).*
- virtual void **close** ()

  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. 2161) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () **const**

  *Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () **const**

  *Is the **Transport** (p. 2161) Connected to its Broker.*

- virtual bool **isClosed** () **const**

  *Has the **Transport** (p. 2161) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () **const**
- virtual void **reconnect** (**const decaf::net::URI** &uri AMQCPP_UNUSED)
- std::string **getName** () **const**
- void **setName** (**const** std::string &name)
- 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)
- virtual bool **isReconnectSupported** () **const**
- virtual bool **isUpdateURIsSupported** () **const**
- virtual void **updateURIs** (bool rebalance AMQCPP_UNUSED, **const decaf::util::List**< **decaf::net::URI** > &uris AMQCPP_UNUSED)

Static Public Member Functions

- static **MockTransport** * **getInstance** ()

6.364.1 Detailed Description

The **MockTransport** (p. 1509) defines a base level **Transport** (p. 2161) class that is intended to be used in place of an a regular protocol **Transport** (p. 2161) suck as TCP.

This **Transport** (p. 2161) assumes that it is the base **Transport** (p. 2161) in the Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p. 2161) defines an Interface **ResponseBuilder** (p. 1784) which must be implemented by any protocol for which the **Transport** (p. 2161) is used to Emulate. The **Transport** (p. 2161) hands off all outbound commands to the **ResponseBuilder** (p. 1784) 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.

6.364.2 Constructor & Destructor Documentation

6.364.2.1 activemq::transport::mock::MockTransport::MockTransport (const Pointer< wireformat::WireFormat > & *wireFormat,* const Pointer< ResponseBuilder > & *responseBuilder*)

6.364.2.2 virtual activemq::transport::mock::MockTransport::~MockTransport ( ) [inline, virtual]

6.364.3 Member Function Documentation

6.364.3.1 virtual void activemq::transport::mock::MockTransport::close ( ) [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. 633).

6.364.3.2 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.

6.364.3.3 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. 2161) listener.

6.364.3.4 static MockTransport* activemq::transport::mock::MockTransport::getInstance ( ) [inline, static]

6.364.3.5 std::string activemq::transport::mock::MockTransport::getName ( ) const [inline]

6.364.3.6 int activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail ( ) const [inline]

6.364.3.7 int activemq::transport::mock::MockTransport::getNumReceivedMessages ( ) const [inline]

6.364.3.8 int activemq::transport::mock::MockTransport::getNumSentKeepAlives ( ) const [inline]

6.364.3.9 int activemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail ( ) const [inline]

6.364.3.10 int activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail ( ) const [inline]

6.364.3.11 int activemq::transport::mock::MockTransport::getNumSentMessages ( ) const [inline]

6.364.3.12 virtual std::string activemq::transport::mock::MockTransport::getRemoteAddress ( ) const [inline, virtual]

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2162).

6.364.3.13 virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener ( ) const [inline, virtual]

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2163).

6.364.3.14 Pointer<wireformat::WireFormat> activemq::transport::mock::MockTransport::getWireFormat ( ) const [inline, virtual]

Gets the currently set WireFormat.

Returns

the current WireFormat object.

Implements **activemq::transport::Transport** (p. 2163).

6.364.3.15 virtual bool activemq::transport::mock::MockTransport::isClosed ( ) const [inline, virtual]

Has the **Transport** (p. 2161) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2161)

Implements **activemq::transport::Transport** (p. 2163).

6.364.3.16 virtual bool activemq::transport::mock::MockTransport::isConnected ( ) const [inline, virtual]

Is the **Transport** (p. 2161) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2163).

6.364.3.17 bool activemq::transport::mock::MockTransport::isFailOnClose ( ) const [inline]

6.364.3.18 bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends ( ) const [inline]

6.364.3.19 bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage ( ) const [inline]

6.364.3.20 bool activemq::transport::mock::MockTransport::isFailOnSendMessage ( ) const [inline]

6.364.3.21 bool activemq::transport::mock::MockTransport::isFailOnStart ( ) const [inline]

6.364.3.22 bool activemq::transport::mock::MockTransport::isFailOnStop ( ) const [inline]

6.364.3.23 virtual bool activemq::transport::mock::MockTransport::isFaultTolerant ( ) const [inline, virtual]

Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2161) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2163).

6.364.3.24 virtual bool activemq::transport::mock::MockTransport::isReconnectSupported ( ) const [inline, virtual]

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.364.3.25 virtual bool activemq::transport::mock::MockTransport::isUpdateURIsSupported ( ) const [inline, virtual]

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.364.3.26 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. 2161) 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. 2164).

6.364.3.27 virtual void activemq::transport::mock::MockTransport::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2164).

6.364.3.28 virtual void activemq::transport::mock::MockTransport::reconnect (const decaf::net::URI &uri *AMQCPP_UNUSED*) [inline, virtual]

6.364.3.29 virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2165).

6.364.3.30 virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & *command,* unsigned int *timeout*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2165).

6.364.3.31 void **activemq::transport::mock::MockTransport::setFailOnClose** (bool *value*) [inline]

6.364.3.32 void **activemq::transport::mock::MockTransport::setFailOnKeepAliveSends** (bool *value*) [inline]

6.364.3.33 void **activemq::transport::mock::MockTransport::setFailOnReceiveMessage** (bool *value*) [inline]

6.364.3.34 void **activemq::transport::mock::MockTransport::setFailOnSendMessage** (bool *value*) [inline]

6.364.3.35 void **activemq::transport::mock::MockTransport::setFailOnStart** (bool *value*) [inline]

6.364.3.36 void **activemq::transport::mock::MockTransport::setFailOnStop** (bool *value*) [inline]

6.364.3.37 void **activemq::transport::mock::MockTransport::setName** (const std::string & *name*) [inline]

6.364.3.38 void **activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail** (int *value*) [inline]

6.364.3.39 void **activemq::transport::mock::MockTransport::setNumReceivedMessages** (int *value*) [inline]

6.364.3.40 void **activemq::transport::mock::MockTransport::setNumSentKeepAlives** (int *value*) [inline]

6.364.3.41 void **activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail** (int *value*) [inline]

6.364.3.42 void **activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail** (int *value*) [inline]

6.364.3.43 void **activemq::transport::mock::MockTransport::setNumSentMessages** (int *value*) [inline]

6.364.3.44 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

6.364.3.45 void **activemq::transport::mock::MockTransport::setResponseBuilder** (const Pointer< ResponseBuilder > & *responseBuilder*) [inline]

Sets the **ResponseBuilder** (p. 1784) 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. 1784) to use from now on.

6.364.3.46 virtual void activemq::transport::mock::MockTransport::setTransportListener (TransportListener * *listener*) [inline, virtual]

Sets the observer of asynchronous events from this transport.

Parameters

listener	the listener of transport events.

Implements **activemq::transport::Transport** (p. 2166).

6.364.3.47 virtual void activemq::transport::mock::MockTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP_UNUSED*) [inline, virtual]

6.364.3.48 virtual void activemq::transport::mock::MockTransport::start ( ) [virtual]

Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.

Exceptions

IOException	if and error occurs while starting the **Transport** (p. 2161).

Implements **activemq::transport::Transport** (p. 2166).

6.364.3.49 virtual void activemq::transport::mock::MockTransport::stop ( ) [virtual]

Stops the **Transport** (p. 2161).

Exceptions

IOException	if an error occurs while stopping the transport.

Implements **activemq::transport::Transport** (p. 2167).

6.364.3.50 virtual void activemq::transport::mock::MockTransport::updateURIs (bool rebalance *AMQCPP_UNUSED*, const decaf::util::List< decaf::net::URI > &uris *AMQCPP_UNUSED*) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransport.h**

6.365 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)

  *Creates a fully configured **Transport** (p. 2161) instance which could be a chain of filters and transports.*
- virtual **Pointer**< **Transport** > **createComposite** (**const decaf::net::URI** &location)

  *Creates a slimed down **Transport** (p. 2161) 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)

  *Creates a slimed down **Transport** (p. 2161) instance which can be used in composite transport instances.*

6.365.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

6.365.2 Constructor & Destructor Documentation

6.365.2.1 virtual activemq::transport::mock::MockTransportFactory::~MockTransportFactory ( ) [inline, virtual]

6.365.3 Member Function Documentation

6.365.3.1 virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::create (const decaf::net::URI & *location*) [virtual]

Creates a fully configured **Transport** (p. 2161) 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. 2168).

6.365.3.2 virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::createComposite (const decaf::net::URI & *location*) [virtual]

Creates a slimed down **Transport** (p. 2161) 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. 2168).

6.365.3.3 virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::doCreateComposite (const decaf::net::URI & *location,* const Pointer< wireformat::WireFormat > & *wireFormat,* const decaf::util::Properties & *properties*) [protected, virtual]

Creates a slimed down **Transport** (p. 2161) 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. 2161).
properties	- Properties to apply to the transport.

Returns

Pointer to a new **Transport** (p. 2161) 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**

6.366 decaf::util::concurrent::Mutex Class Reference

Mutex (p. 1519) 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** ()
- **Mutex** (**const** std::string &name)
- virtual ~**Mutex** ()
- std::string **getName** () **const**
- std::string **toString** () **const**
- virtual void **lock** ()

  Locks the object.
- virtual bool **tryLock** ()

  *Attempts to **Lock** (p. 1304) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()

  Unlocks the object.
- virtual void **wait** ()

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)

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

- virtual void **wait** (long long millisecs, int nanos)

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

- virtual void **notify** ()

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

- virtual void **notifyAll** ()

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

6.366.1 Detailed Description

Mutex (p. 1519) 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

6.366.2 Constructor & Destructor Documentation

6.366.2.1 decaf::util::concurrent::Mutex::Mutex ( )

6.366.2.2 decaf::util::concurrent::Mutex::Mutex (const std::string & *name*)

6.366.2.3 virtual decaf::util::concurrent::Mutex::~Mutex ( ) [virtual]

6.366.3 Member Function Documentation

6.366.3.1 std::string decaf::util::concurrent::Mutex::getName ( ) const

6.366.3.2 virtual void decaf::util::concurrent::Mutex::lock ( ) [virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

Referenced by decaf::util::StlQueue< T >::lock(), decaf::util::StlMap< std::string, cms::Topic * >::lock(), decaf::util::AbstractCollection< cms::Connection * >::lock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::lock().

6.366.3.3 virtual void decaf::util::concurrent::Mutex::notify ( ) [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. 2045) Object.
RuntimeException	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2047).

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::clear(), decaf::util::StlQueue< T >::notify(), decaf::util::StlMap< std::string, cms::Topic * >::notify(), decaf::util::AbstractCollection< cms::Connection * >::notify(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::notify(), decaf::util::concurrent::LinkedBlockingQueue< E >::offer(), decaf::util::concurrent::LinkedBlockingQueue< E >::poll(), decaf::util::concurrent::LinkedBlockingQueue< E >::put(), and decaf::util::concurrent::LinkedBlockingQueue< E >::take().

6.366.3.4 virtual void decaf::util::concurrent::Mutex::notifyAll ( ) [virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.
RuntimeException	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2048).

Referenced by decaf::util::StlQueue< T >::notifyAll(), decaf::util::StlMap< std::string, cms::Topic * >::notifyAll(), decaf::util::AbstractCollection< cms::Connection * >::notifyAll(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::notifyAll().

6.366.3.5 std::string decaf::util::concurrent::Mutex::toString ( ) const

6.366.3.6 virtual bool decaf::util::concurrent::Mutex::tryLock ( ) [virtual]

Attempts to **Lock** (p. 1304) 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. 2048).

Referenced by decaf::util::StlQueue< T >::tryLock(), decaf::util::StlMap< std::string, cms::Topic * >::tryLock(), decaf::util::AbstractCollection< cms::Connection * >::tryLock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::tryLock().

6.366.3.7 virtual void decaf::util::concurrent::Mutex::unlock ( ) [virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

Referenced by decaf::util::StlQueue< T >::unlock(), decaf::util::StlMap< std::string, cms::Topic * >::unlock(), decaf::util::AbstractCollection< cms::Connection * >::unlock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::unlock().

6.366.3.8 virtual void decaf::util::concurrent::Mutex::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::offer(), decaf::util::concurrent::LinkedBlockingQueue< E >::poll(), decaf::util::concurrent::LinkedBlockingQueue< E >::put(), decaf::util::concurrent::LinkedBlockingQueue< E >::take(), decaf::util::StlQueue< T >::wait(), decaf::util::StlMap< std::string, cms::Topic * >::wait(), decaf::util::AbstractCollection< cms::Connection * >::wait(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait(), and decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::wait().

6.366.3.9 virtual void decaf::util::concurrent::Mutex::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.366.3.10 virtual void decaf::util::concurrent::Mutex::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Mutex.h**

6.367 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**

6.367.1 Constructor & Destructor Documentation

6.367.1.1 decaf::util::concurrent::MutexHandle::MutexHandle ( ) [inline]

6.367.1.2 decaf::util::concurrent::MutexHandle::~MutexHandle ( ) [inline]

6.367.1.3 decaf::util::concurrent::MutexHandle::MutexHandle ( ) [inline]

6.367.1.4 decaf::util::concurrent::MutexHandle::~MutexHandle ( ) [inline]

6.367.2 Field Documentation

6.367.2.1 volatile long long decaf::util::concurrent::MutexHandle::lock_count

6.367.2.2 volatile long long decaf::util::concurrent::MutexHandle::lock_owner

6.367.2.3 CRITICAL_SECTION decaf::util::concurrent::MutexHandle::mutex

6.367.2.4 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**

6.368 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.

6.368.1 Member Function Documentation

6.368.1.1 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.

6.368.1.2 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.

6.368.1.3 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.

6.368.1.4 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.

6.368.1.5 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**

6.369 decaf::internal::net::Network Class Reference

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

`#include <src/main/decaf/internal/net/Network.h>`

Public Member Functions

- virtual ~**Network** ()
- **decaf::util::concurrent::Mutex** * **getRuntimeLock** ()

  *Gets a pointer to the **Network** (p. 1525) Runtime's Lock object, this can be used by **Network** (p. 1525) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 1525) layer, etc.*

- void **addNetworkResource** (**decaf::internal::util::Resource** ∗value)

  *Adds a Resource to the **Network** (p. 1525) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 1525) Runtime are destroyed.*

- template<typename T >
  void **addAsResource** (T ∗value)
- void **addShutdownTask** (**decaf::lang::Runnable** ∗task)

  *Register a Runnable to be called when the **Network** (p. 1525) Runtime is shutdown to provide a chance to cleanup any data or references that could cause problems should the **Network** (p. 1525) Runtime be re-initialized.*

Static Public Member Functions

- static **Network** ∗ **getNetworkRuntime** ()

  *Gets the one and only instance of the **Network** (p. 1525) class, if this is called before the **Network** (p. 1525) layer has been initialized or after it has been shutdown then an IllegalStateException is thrown.*

- static void **initializeNetworking** ()

  Initialize the Networking layer.

- static void **shutdownNetworking** ()

  *Shutdown the **Network** (p. 1525) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.*

Protected Member Functions

- **Network** ()

6.369.1 Detailed Description

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

Since

1.0

6.369.2 Constructor & Destructor Documentation

6.369.2.1 decaf::internal::net::Network::Network ( ) [protected]

6.369.2.2 virtual decaf::internal::net::Network::∼Network ( ) [virtual]

6.369.3 Member Function Documentation

6.369.3.1 template<typename T > void decaf::internal::net::Network::addAsResource (T ∗ *value*) [inline]

6.369.3.2 void decaf::internal::net::Network::addNetworkResource (decaf::internal::util::Resource ∗ *value*)

Adds a Resource to the **Network** (p. 1525) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 1525) Runtime are destroyed.

Parameters

value	The Resource to add to the **Network** (p. 1525) Runtime.

Exceptions

NullPointerException	if the Resource value passed is null.

6.369.3.3 void decaf::internal::net::Network::addShutdownTask (decaf::lang::Runnable * *task*)

Register a Runnable to be called when the **Network** (p. 1525) Runtime is shutdown to provide a chance to cleanup any data or references that could cause problems should the **Network** (p. 1525) Runtime be re-initialized.

The Runnable pointer ownership is transfered to the NetworkRuntime to guarantee the timing of resource cleanup.

The cleanup tasks are run at a critical time in the Shutdown process and should be as simple as possible and make every attempt to no throw any exceptions. If an exception is thrown it is ignored and processing of the next task is started.

The tasks should not assume that any **Network** (p. 1525) resources are still available and should execute as quickly as possible.

Parameters

task	Pointer to a Runnable object that will now be owned by the **Network** (p. 1525) Runtime.

6.369.3.4 static Network* decaf::internal::net::Network::getNetworkRuntime ( ) [static]

Gets the one and only instance of the **Network** (p. 1525) class, if this is called before the **Network** (p. 1525) layer has been initialized or after it has been shutdown then an IllegalStateException is thrown.

Returns

pointer to the **Network** (p. 1525) runtime for the Decaf library.

6.369.3.5 decaf::util::concurrent::Mutex* decaf::internal::net::Network::getRuntimeLock ( )

Gets a pointer to the **Network** (p. 1525) Runtime's Lock object, this can be used by **Network** (p. 1525) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 1525) layer, etc.

The pointer returned is owned by the **Network** (p. 1525) runtime and should not be deleted or copied by the caller.

Returns

a pointer to the **Network** (p. 1525) Runtime's single Lock instance.

6.369.3.6 static void decaf::internal::net::Network::initializeNetworking ( ) [static]

Initialize the Networking layer.

6.369.3.7 static void decaf::internal::net::Network::shutdownNetworking ( ) [static]

Shutdown the **Network** (p. 1525) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**Network.h**

6.370 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **NetworkBridgeFilter** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

6.370.1 Constructor & Destructor Documentation

6.370.1.1 activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter ( )

6.370.1.2 virtual activemq::commands::NetworkBridgeFilter::~NetworkBridgeFilter ( ) [virtual]

6.370.2 Member Function Documentation

6.370.2.1 virtual NetworkBridgeFilter* activemq::commands::NetworkBridgeFilter::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.370.2.2 virtual void activemq::commands::NetworkBridgeFilter::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. 879).

6.370.2.3 virtual bool activemq::commands::NetworkBridgeFilter::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.370.2.4 virtual unsigned char activemq::commands::NetworkBridgeFilter::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.370.2.5 virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId ( ) const [virtual]

6.370.2.6 virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId ( ) [virtual]

6.370.2.7 virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL ( ) const [virtual]

6.370.2.8 virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId (const Pointer< BrokerId > & *networkBrokerId*) [virtual]

6.370.2.9 virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL (int *networkTTL*) [virtual]

6.370.2.10 virtual std::string activemq::commands::NetworkBridgeFilter::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.370.3 Field Documentation

6.370.3.1 const unsigned char activemq::commands::NetworkBridgeFilter::ID_NETWORKBRIDGEFILTER = 91 [static]

6.370.3.2 Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId [protected]

6.370.3.3 int activemq::commands::NetworkBridgeFilter::networkTTL [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**NetworkBridgeFilter.h**

6.371 activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1530).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/NetworkBridgeFilterMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual ~**NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.371.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1530).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.371.2 Constructor & Destructor Documentation

6.371.2.1 activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller ( ) [inline]

6.371.2.2 virtual activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::~NetworkBridgeFilterMarshaller ( ) [inline, virtual]

6.371.3 Member Function Documentation

6.371.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.371.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.371.3.3 virtual void activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.371.3.4 virtual void activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.371.3.5 virtual int activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.371.3.6 virtual void activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.371.3.7 virtual void activemq::wireformat::openwire::marshal::generated::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**NetworkBridgeFilterMarshaller.h**

6.372 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 ()

6.372.1 Constructor & Destructor Documentation

6.372.1.1 decaf::net::NoRouteToHostException::NoRouteToHostException ( ) throw () [inline]

Default Constructor.

6.372.1.2 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

6.372.1.3 decaf::net::NoRouteToHostException::NoRouteToHostException (const NoRouteToHostException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.372.1.4 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

6.372.1.5 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.

6.372.1.6 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

6.372.1.7 virtual decaf::net::NoRouteToHostException::~NoRouteToHostException ( ) throw () [inline, virtual]

6.372.2 Member Function Documentation

6.372.2.1 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. 1916).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**NoRouteToHostException.h**

6.373 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 ()

6.373.1 Constructor & Destructor Documentation

6.373.1.1 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( ) throw () [inline]

Default Constructor.

6.373.1.2 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

6.373.1.3 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const NoSuchAlgorithmException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.373.1.4 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

6.373.1.5 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.

6.373.1.6 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

6.373.1.7 virtual decaf::security::NoSuchAlgorithmException::~NoSuchAlgorithmException ( ) throw ()

[inline, virtual]

6.373.2 Member Function Documentation

6.373.2.1 virtual NoSuchAlgorithmException* decaf::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. 1081).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**NoSuchAlgorithmException.h**

6.374 decaf::util::NoSuchElementException Class Reference

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

Inheritance diagram for decaf::util::NoSuchElementException:

Public Member Functions

- **NoSuchElementException** () throw ()

  Default Constructor.
- **NoSuchElementException** (**const decaf::lang::exceptions::RuntimeException** &ex) throw ()

  Conversion Constructor from some other Exception.
- **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 occured.
- virtual **NoSuchElementException** * **clone** () **const**

  Clones this exception.
- virtual ~**NoSuchElementException** () throw ()

6.374.1 Constructor & Destructor Documentation

6.374.1.1 decaf::util::NoSuchElementException::NoSuchElementException ( ) throw ()

Default Constructor.

6.374.1.2 decaf::util::NoSuchElementException::NoSuchElementException (const decaf::lang::exceptions::RuntimeException & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex	The Exception whose data is to be copied into this one.

6.374.1.3 decaf::util::NoSuchElementException::NoSuchElementException (const NoSuchElementException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The Exception whose data is to be copied into this one.

6.374.1.4 decaf::util::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

6.374.1.5 decaf::util::NoSuchElementException::NoSuchElementException (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.

6.374.1.6 decaf::util::NoSuchElementException::NoSuchElementException (const char * *file,* const int *lineNumber,* const char * *msg, ...*) throw () [inline]

Constructor - Initializes the file name and line number where this message occured.

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

6.374.1.7 virtual decaf::util::NoSuchElementException::~NoSuchElementException ( ) throw ()

[virtual]

6.374.2 Member Function Documentation

6.374.2.1 virtual NoSuchElementException* decaf::util::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 instance that is a copy of this one.

Reimplemented from **decaf::lang::exceptions::RuntimeException** (p. 1796).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**NoSuchElementException.h**

6.375 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 occured.
- virtual **NoSuchProviderException** * **clone** () **const**

  Clones this exception.
- virtual ~**NoSuchProviderException** () throw ()

6.375.1 Constructor & Destructor Documentation

6.375.1.1 decaf::security::NoSuchProviderException::NoSuchProviderException ( ) throw () [inline]

Default Constructor.

6.375.1.2 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

6.375.1.3 decaf::security::NoSuchProviderException::NoSuchProviderException (const NoSuchProviderException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.375.1.4 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

6.375.1.5 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.

6.375.1.6 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 occured.

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

6.375.1.7 virtual decaf::security::NoSuchProviderException::~NoSuchProviderException ( ) throw () [inline, virtual]

6.375.2 Member Function Documentation

6.375.2.1 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. 1081).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**NoSuchProviderException.h**

6.376 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. 990).*
- **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 occured.
- virtual **NullPointerException** * **clone** () **const**
  Clones this exception.
- virtual ~**NullPointerException** () throw ()

6.376.1 Constructor & Destructor Documentation

6.376.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException ( ) throw () [inline]

Default Constructor.

6.376.1.2 decaf::lang::exceptions::NullPointerException::NullPointerException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.376.1.3 decaf::lang::exceptions::NullPointerException::NullPointerException (const NullPointerException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.376.1.4 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

6.376.1.5 decaf::lang::exceptions::NullPointerException::NullPointerException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.376.1.6 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 occured.

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

6.376.1.7 virtual decaf::lang::exceptions::NullPointerException::~NullPointerException ( ) throw ()

[inline, virtual]

6.376.2 Member Function Documentation

6.376.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NullPointerException.h**

6.377 decaf::lang::Number Class Reference

The abstract class **Number** (p. 1543) is the superclass of classes **Byte** (p. 476), **Double** (p. 956), **Float** (p. 1040), **Integer** (p. 1161), **Long** (p. 1338), and **Short** (p. 1858).

`#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.

6.377.1 Detailed Description

The abstract class **Number** (p. 1543) is the superclass of classes **Byte** (p. 476), **Double** (p. 956), **Float** (p. 1040), **Integer** (p. 1161), **Long** (p. 1338), and **Short** (p. 1858).

Subclasses of **Number** (p. 1543) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

Since

1.0

6.377.2 Constructor & Destructor Documentation

6.377.2.1 virtual decaf::lang::Number::~Number ( ) [inline, virtual]

6.377.3 Member Function Documentation

6.377.3.1 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.

Reimplemented in **decaf::lang::Double** (p. 959), **decaf::lang::Float** (p. 1042), **decaf::lang::Integer** (p. 1164), **decaf::lang::Byte** (p. 478), **decaf::lang::Long** (p. 1341), **decaf::lang::Short** (p. 1860), and **decaf::lang::Character** (p. 595).

6.377.3.2 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. 370), **decaf::lang::Double** (p. 960), **decaf::lang::Float** (p. 1043), **decaf::lang::Integer** (p. 1165), **decaf::lang::Byte** (p. 479), **decaf::lang::Long** (p. 1342), **decaf::lang::Short** (p. 1861), and **decaf::lang::Character** (p. 596).

6.377.3.3 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::lang::Double** (p. 961), **decaf::lang::Float** (p. 1045), **decaf::lang::Integer** (p. 1166), **decaf::lang::Byte** (p. 480), **decaf::lang::Long** (p. 1343), **decaf::util::concurrent::atomic::AtomicInteger** (p. 370), **decaf::lang::Short** (p. 1861), and **decaf::lang::Character** (p. 596).

6.377.3.4 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::lang::Double** (p. 961), **decaf::lang::Float** (p. 1045), **decaf::lang::Integer** (p. 1167), **decaf::lang::Byte** (p. 480), **decaf::lang::Long** (p. 1343), **decaf::lang::Short** (p. 1862), **decaf::lang::Character** (p. 597), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 372).

6.377.3.5 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::lang::Double** (p. 962), **decaf::lang::Float** (p. 1046), **decaf::lang::Integer** (p. 1167), **decaf::lang::Byte** (p. 480), **decaf::lang::Long** (p. 1343), **decaf::lang::Short** (p. 1862), **decaf::lang::Character** (p. 597), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 372).

6.377.3.6 virtual short decaf::lang::Number::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

Reimplemented in **decaf::lang::Double** (p. 964), **decaf::lang::Float** (p. 1048), **decaf::lang::Integer** (p. 1171), **decaf::lang::Byte** (p. 482), **decaf::lang::Long** (p. 1348), **decaf::lang::Short** (p. 1864), and **decaf::lang::Character** (p. 599).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Number.h**

6.378 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. 990).*
- **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 occured.
- virtual **NumberFormatException** ∗ **clone** () **const**
  Clones this exception.
- virtual ∼**NumberFormatException** () throw ()

6.378.1 Constructor & Destructor Documentation

6.378.1.1 decaf::lang::exceptions::NumberFormatException::NumberFormatException ( ) [inline]

Default Constructor.

Referenced by clone().

6.378.1.2 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.378.1.3 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const NumberFormatException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.378.1.4 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().

6.378.1.5 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.378.1.6 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 occured.

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().

6.378.1.7 virtual decaf::lang::exceptions::NumberFormatException::~NumberFormatException ( ) throw () [inline, virtual]

6.378.2 Member Function Documentation

6.378.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

References NumberFormatException().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NumberFormatException.h**

6.379 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** () throw ()

6.379.1 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. 1547) s.

Since

1.0

6.379.2 Constructor & Destructor Documentation

6.379.2.1 virtual cms::ObjectMessage::~ObjectMessage ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**ObjectMessage.h**

6.380 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference

Provides an SSLContext that wraps the OpenSSL API.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLContextSpi:

Public Member Functions

- **OpenSSLContextSpi** ()
- virtual ~**OpenSSLContextSpi** ()
- virtual void **providerInit** (**security::SecureRandom** *random)

  Perform the initialization of this Context.

  Parameters

random	*Pointer to an instance of a secure random number generator.*

  Exceptions

NullPointerException	*if the SecureRandom instance is NULL.*
KeyManagementException	*if an error occurs while initializing the context.*

- virtual **decaf::net::SocketFactory** * **providerGetSocketFactory** ()

  *Returns a **SocketFactory** (p. 1916) instance that can be used to create new **SSLSocket** (p. 1947) objects.*
  *The **SocketFactory** (p. 1916) is owned by the Service Provider and should not be destroyed by the caller.*

  Returns

  ***SocketFactory** (p. 1916) instance that can be used to create new SSLSockets.*

  Exceptions

IllegalStateException	*if the **SSLContextSpi** (p. 1936) object requires initialization but has not been initialized yet.*

- virtual **decaf::net::ServerSocketFactory** * **providerGetServerSocketFactory** ()

  *Returns a **ServerSocketFactory** (p. 1823) instance that can be used to create new **SSLServerSocket** (p. 1941) objects.*
  *The **ServerSocketFactory** (p. 1823) is owned by the Service Provider and should not be destroyed by the caller.*

Returns

SocketFactory *(p. 1916) instance that can be used to create new SSLServerSockets.*

Exceptions

IllegalStateException	*if the **SSLContextSpi** (p. 1936) object requires initialization but has not been initialized yet.*

Friends

- class **OpenSSLSocket**
- class **OpenSSLSocketFactory**

6.380.1 Detailed Description

Provides an SSLContext that wraps the OpenSSL API.

Since

1.0

6.380.2 Constructor & Destructor Documentation

6.380.2.1 decaf::internal::net::ssl::openssl::OpenSSLContextSpi::OpenSSLContextSpi ( )

6.380.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLContextSpi::~OpenSSLContextSpi ( ) [virtual]

6.380.3 Member Function Documentation

6.380.3.1 virtual decaf::net::ServerSocketFactory* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetServerSocketFactory ( ) [virtual]

Returns a **ServerSocketFactory** (p. 1823) instance that can be used to create new **SSLServerSocket** (p. 1941) objects.

The **ServerSocketFactory** (p. 1823) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 1916) instance that can be used to create new SSLServerSockets.

Exceptions

IllegalStateException	if the **SSLContextSpi** (p. 1936) object requires initialization but has not been initialized yet.

Implements **decaf::net::ssl::SSLContextSpi** (p. 1937).

6.380.3.2 virtual decaf::net::SocketFactory* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetSocketFactory ( ) [virtual]

Returns a **SocketFactory** (p. 1916) instance that can be used to create new **SSLSocket** (p. 1947) objects.

The **SocketFactory** (p. 1916) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 1916) instance that can be used to create new SSLSockets.

Exceptions

IllegalStateException	if the **SSLContextSpi** (p. 1936) object requires initialization but has not been initialized yet.

Implements **decaf::net::ssl::SSLContextSpi** (p. 1937).

6.380.3.3 virtual void decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerInit (security::SecureRandom * *random*) [virtual]

Perform the initialization of this Context.

Parameters

random	Pointer to an instance of a secure random number generator.

Exceptions

NullPointerException	if the SecureRandom instance is NULL.
KeyManagementException	if an error occurs while initializing the context.

Implements **decaf::net::ssl::SSLContextSpi** (p. 1938).

6.380.4 Friends And Related Function Documentation

6.380.4.1 friend class OpenSSLSocket [friend]

6.380.4.2 friend class OpenSSLSocketFactory [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLContextSpi.h**

6.381 decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference

Container class for parameters that are Common to OpenSSL socket classes.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h>`

Public Member Functions

- virtual ~**OpenSSLParameters** ()
- bool **getNeedClientAuth** () **const**
- void **setNeedClientAuth** (bool value)
- bool **getWantClientAuth** () **const**
- void **setWantClientAuth** (bool value)
- bool **getUseClientMode** () **const**
- void **setUseClientMode** (bool value)
- std::vector< std::string > **getSupportedCipherSuites** () **const**
- std::vector< std::string > **getSupportedProtocols** () **const**

- std::vector< std::string > **getEnabledCipherSuites** () **const**
- void **setEnabledCipherSuites** (**const** std::vector< std::string > &suites)
- std::vector< std::string > **getEnabledProtocols** () **const**
- void **setEnabledProtocols** (**const** std::vector< std::string > &protocols)
- **OpenSSLParameters** * **clone** () **const**

  Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL_CTX as this object's.

6.381.1 Detailed Description

Container class for parameters that are Common to OpenSSL socket classes.

Since

1.0

6.381.2 Constructor & Destructor Documentation

6.381.2.1 virtual decaf::internal::net::ssl::openssl::OpenSSLParameters::~OpenSSLParameters ( ) [virtual]

6.381.3 Member Function Documentation

6.381.3.1 OpenSSLParameters* decaf::internal::net::ssl::openssl::OpenSSLParameters::clone ( ) const

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL_CTX as this object's.

6.381.3.2 std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledCipherSuites ( ) const

6.381.3.3 std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledProtocols ( ) const

6.381.3.4 bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getNeedClientAuth ( ) const [inline]

6.381.3.5 std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedCipherSuites ( ) const

6.381.3.6 std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedProtocols ( ) const

6.381.3.7 bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getUseClientMode ( ) const [inline]

6.381.3.8 bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getWantClientAuth ( ) const [inline]

6.381.3.9 void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledCipherSuites (const std::vector< std::string > & *suites*)

6.381.3.10 void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledProtocols (const std::vector< std::string > & *protocols*)

6.381.3.11 void decaf::internal::net::ssl::openssl::OpenSSLParameters::setNeedClientAuth (bool *value*) [inline]

6.381.3.12 void decaf::internal::net::ssl::openssl::OpenSSLParameters::setUseClientMode (bool *value*) [inline]

6.381.3.13 void decaf::internal::net::ssl::openssl::OpenSSLParameters::setWantClientAuth (bool *value*) [inline]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLParameters.h**

6.382 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference

SSLServerSocket based on OpenSSL library code.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocket:

Public Member Functions

- **OpenSSLServerSocket** (**OpenSSLParameters** ∗parameters)
- virtual ∼**OpenSSLServerSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () **const**

  *Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 1941). Normally not all of these cipher suites will be enabled on the **Socket** (p. 1900).*

  Returns

  a vector containing the names of all the supported cipher suites.
- virtual std::vector< std::string > **getSupportedProtocols** () **const**

  *Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 1941) instance.*

  Returns

  a vector containing the names of all the supported protocols.
- virtual std::vector< std::string > **getEnabledCipherSuites** () **const**

  *Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 1941).*

  Returns

  vector of the names of all enabled Cipher Suites.
- virtual void **setEnabledCipherSuites** (**const** std::vector< std::string > &suites)

  *Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 1941) connection.*
  Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

  Parameters

suites	*An Vector of names for all the Cipher Suites that are to be enabled.*

  Exceptions

IllegalArgumentException	*if the vector is empty or one of the names is invalid.*

- virtual std::vector< std::string > **getEnabledProtocols** () **const**

  *Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 1941).*

Returns

vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (**const** std::vector< std::string > &protocols)

*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 1941) connection.*
Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols	*An Vector of names for all the Protocols that are to be enabled.*

Exceptions

IllegalArgumentException	*if the vector is empty or one of the names is invalid.*

- virtual bool **getWantClientAuth** () **const**

Returns

*true if the **Socket** (p. 1900) request client Authentication.*

- virtual void **setWantClientAuth** (bool value)

*Sets whether or not this **Socket** (p. 1900) will request Client Authentication.*
*If set to true the **Socket** (p. 1900) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.*

Parameters

value	*Whether the server socket should request client authentication.*

- virtual bool **getNeedClientAuth** () **const**

Returns

*true if the **Socket** (p. 1900) requires client Authentication.*

- virtual void **setNeedClientAuth** (bool value)

*Sets whether or not this **Socket** (p. 1900) will require Client Authentication.*
*If set to true the **Socket** (p. 1900) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.*

Parameters

value	*Whether the server socket should require client authentication.*

- virtual **decaf::net::Socket** * **accept** ()

*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 1816), the caller blocks until a connection is made.*
*If the SO_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 1932) if the operation times out.*

Returns

*a new **Socket** (p. 1900) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.*

Exceptions

IOException	*if an I/O error occurs while binding the socket.*
SocketException (p. 1915)	*if an error occurs while blocking on the accept call.*
SocketTimeoutException (p. 1932)	*if the SO_TIMEOUT option was used and the accept timed out.*

6.382.1 Detailed Description

SSLServerSocket based on OpenSSL library code.

Since

1.0

6.382.2 Constructor & Destructor Documentation

6.382.2.1 decaf::internal::net::ssl::openssl::OpenSSLServerSocket::OpenSSLServerSocket (OpenSSLParameters * *parameters*)

6.382.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocket::~OpenSSLServerSocket ( ) [virtual]

6.382.3 Member Function Documentation

6.382.3.1 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLServerSocket::accept ( ) [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 1816), the caller blocks until a connection is made.

If the SO_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 1932) if the operation times out.

Returns

a new **Socket** (p. 1900) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

Exceptions

IOException	if an I/O error occurs while binding the socket.
SocketException *(p. 1915)*	if an error occurs while blocking on the accept call.
SocketTimeoutException *(p. 1932)*	if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented from **decaf::net::ServerSocket** (p. 1819).

6.382.3.2 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledCipherSuites ( ) const [virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 1941).

Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1943).

6.382.3.3 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledProtocols ( ) const [virtual]

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 1941).

Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1943).

6.382.3.4 virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getNeedClientAuth ( ) const [virtual]

Returns

true if the **Socket** (p. 1900) requires client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1944).

6.382.3.5 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedCipherSuites ( ) const [virtual]

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 1941).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 1900).

Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1944).

6.382.3.6 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedProtocols ( ) const [virtual]

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 1941) instance.

Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1944).

6.382.3.7 virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getWantClientAuth ( ) const [virtual]

Returns

true if the **Socket** (p. 1900) request client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1944).

6.382.3.8 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledCipherSuites (const std::vector< std::string > & *suites*) [virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 1941) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites	An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1944).

6.382.3.9 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledProtocols (const std::vector< std::string > & *protocols*) [virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 1941) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols	An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1945).

6.382.3.10 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setNeedClientAuth (bool *value*) [virtual]

Sets whether or not this **Socket** (p. 1900) will require Client Authentication.

If set to true the **Socket** (p. 1900) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

Parameters

value	Whether the server socket should require client authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1945).

6.382.3.11 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setWantClientAuth (bool *value*) [virtual]

Sets whether or not this **Socket** (p. 1900) will request Client Authentication.

If set to true the **Socket** (p. 1900) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

Parameters

value	Whether the server socket should request client authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 1945).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLServerSocket.h**

6.383 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory:

Public Member Functions

- **OpenSSLServerSocketFactory** (**OpenSSLContextSpi** ∗parent)
- virtual ∼**OpenSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket** ∗ **createServerSocket** ()

  *Create a new **ServerSocket** (p. 1816) that is unbound.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** ∗ **createServerSocket** (int port)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** ∗ **createServerSocket** (int port, int backlog)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will use the specified connection backlog setting.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*
backlog	*The number of pending connect request the **ServerSocket** (p. 1816) can queue.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

  Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual **decaf::net::ServerSocket** ∗ **createServerSocket** (int port, int backlog, **const decaf::net::InetAddress** ∗address)

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
  *The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 1816) will listen on all interfaces.*

  Parameters

port	*The port to bind the **ServerSocket** (p. 1816) to.*
backlog	*The number of pending connect request the **ServerSocket** (p. 1816) can queue.*
address	*The address of the interface on the local machine to bind to.*

  Returns

  *new **ServerSocket** (p. 1816) pointer that is owned by the caller.*

Exceptions

IOException	*if the **ServerSocket** (p. 1816) cannot be created for some reason.*

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

  Returns the list of cipher suites which are enabled by default.
  Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

  Returns

  an STL vector containing the list of cipher suites enabled by default.

  See also

  getSupportedCipherSuites() (p. 1947)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

  Returns the names of the cipher suites which could be enabled for use on an SSL connection.
  Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

  Returns

  an STL vector containing the list of supported cipher suites.

  See also

  getDefaultCipherSuites() (p. 1947)

6.383.1 Detailed Description

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

Since

1.0

6.383.2 Constructor & Destructor Documentation

6.383.2.1 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::OpenSSLServerSocketFactory (OpenSSLContextSpi * *parent*)

6.383.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::~OpenSSLServerSocketFactory ( ) [virtual]

6.383.3 Member Function Documentation

6.383.3.1 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket ( ) [virtual]

Create a new **ServerSocket** (p. 1816) that is unbound.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 1824).

6.383.3.2 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket (int *port*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1824).

6.383.3.3 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket (int *port,* int *backlog*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will use the specified connection backlog setting.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1825).

6.383.3.4 virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket (int *port,* int *backlog,* const decaf::net::InetAddress * *address*) [virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 1816) will listen on all interfaces.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.

address	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 1825).

6.383.3.5 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getDefaultCipherSuites ( ) [virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 1947)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 1947).

6.383.3.6 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getSupportedCipherSuites ( ) [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 1947)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 1947).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLServerSocketFactory.h**

6.384 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocket:

Public Member Functions

- **OpenSSLSocket** (**OpenSSLParameters** ∗parameters)
- **OpenSSLSocket** (**OpenSSLParameters** ∗parameters, **const decaf::net::InetAddress** ∗address, int port)
- **OpenSSLSocket** (**OpenSSLParameters** ∗parameters, **const decaf::net::InetAddress** ∗address, int port, **const decaf::net::InetAddress** ∗localAddress, int localPort)
- **OpenSSLSocket** (**OpenSSLParameters** ∗parameters, **const** std::string &host, int port)
- **OpenSSLSocket** (**OpenSSLParameters** ∗parameters, **const** std::string &host, int port, **const decaf::net::-InetAddress** ∗localAddress, int localPort)
- virtual ∼**OpenSSLSocket** ()
- virtual void **connect** (**const** std::string &host, int port, int timeout)

  Connects to the specified destination, with a specified timeout value.
  *If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeout-Exception** (p. 1932) is thrown. A timeout value of zero is treated as an inifinite timeout.*

  Parameters

host	*The host name or IP address of the remote host to connect to.*
port	*The port on the remote host to connect to.*
timeout	*The number of Milliseconds to wait before treating the connection as failed.*

  Exceptions

IOException	*Thrown if a failure occurred in the connect.*
SocketTimeoutException (p. 1932)	*if the timeout for connection is exceeded.*
IllegalArguementException	*if the timeout value is negative or the endpoint is invalid.*

- virtual void **close** ()

  *Closes the **Socket** (p. 1900).*
  *Once closed a **Socket** (p. 1900) cannot be connected or otherwise operated upon, a new **Socket** (p. 1900) instance must be created.*

  Exceptions

IOException	*if an I/O error occurs while closing the **Socket** (p. 1900).*

- virtual **decaf::io::InputStream** ∗ **getInputStream** ()

  Gets the InputStream for this socket if its connected.
  *The pointer returned is the property of the associated **Socket** (p. 1900) and should not be deleted by the caller.*
  When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.
  *Closing the InputStream will also close the underlying **Socket** (p. 1900).*

  Returns

  The InputStream for this socket.

  Exceptions

IOException	*if an error occurs during creation of the InputStream, also if the **Socket** (p. 1900) is not connected or the input has been shutdown previously.*

- virtual **decaf::io::OutputStream** ∗ **getOutputStream** ()

  Gets the OutputStream for this socket if it is connected.

*The pointer returned is the property of the **Socket** (p. 1900) instance and should not be deleted by the caller.*
*Closing the returned **Socket** (p. 1900) will also close the underlying **Socket** (p. 1900).*

Returns

the OutputStream for this socket.

Exceptions

IOException	*if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 1900) is closed or the output has been shutdown previously.*

- virtual void **shutdownInput** ()

  Shuts down the InputStream for this socket essentially marking it as EOF.
  The stream returns EOF for any calls to read after this method has been called.

  Exceptions

IOException	*if an I/O error occurs while performing this operation.*

- virtual void **shutdownOutput** ()

  Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OuputStream::write will throw an IOException.

  Exceptions

IOException	*if an I/O error occurs while performing this operation.*

- virtual void **setOOBInline** (bool value)

  Sets the value of the OOBINLINE for this socket, by default this option is disabled.
  *If enabled the urgent data is read inline on the **Socket** (p. 1900)'s InputStream, no notification is give.*

  Returns

  true if OOBINLINE is enabled, false otherwise.

  Exceptions

SocketException (p. 1915)	*if an error is encountered while performing this operation.*

- virtual void **sendUrgentData** (int data)

  *Sends on byte of urgent data to the **Socket** (p. 1900).*

  Parameters

data	*The value to write as urgent data, only the lower eight bits are sent.*

  Exceptions

IOException	*if an I/O error occurs while performing this operation.*

- virtual std::vector< std::string > **getSupportedCipherSuites** () **const**

  *Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 1947).*
  *Normally not all of these cipher suites will be enabled on the **Socket** (p. 1900).*

  Returns

  a vector containing the names of all the supported cipher suites.

- virtual std::vector< std::string > **getSupportedProtocols** () **const**

  *Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 1947) instance.*

  Returns

  a vector containing the names of all the supported protocols.

- virtual std::vector< std::string > **getEnabledCipherSuites** () **const**

  *Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 1900).*

  Returns

  vector of the names of all enabled Cipher Suites.

- virtual void **setEnabledCipherSuites** (**const** std::vector< std::string > &suites)

  *Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 1900) connection.*
  Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites	*An Vector of names for all the Cipher Suites that are to be enabled.*

Exceptions

IllegalArgumentException	*if the vector is empty or one of the names is invalid.*

- virtual std::vector< std::string > **getEnabledProtocols** () **const**

  *Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 1900).*

  Returns

  vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (**const** std::vector< std::string > &protocols)

  *Sets the Protocols that are to be enabled on the SSL **Socket** (p. 1900) connection.*
  Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

  Parameters

protocols	*An Vector of names for all the Protocols that are to be enabled.*

  Exceptions

IllegalArgumentException	*if the vector is empty or one of the names is invalid.*

- virtual void **startHandshake** ()

  Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.
  When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not require to support multiple handshakes and can throw an IOException to indicate an error.

  Exceptions

IOException	*if an I/O error occurs while performing the Handshake*

- virtual void **setUseClientMode** (bool value)

  Determines the mode that the socket uses when a handshake is initiated, client or server.
  *This method must be called prior to any handshake attempts on this **Socket** (p. 1900), once a handshake has be initiated this socket remains the the set mode; client or server, for the life of this object.*

  Parameters

value	*The mode setting, true for client or false for server.*

  Exceptions

IllegalArguementException	*if the handshake process has begun and mode is lcoked.*

- virtual bool **getUseClientMode** () **const**

  *Gets whether this **Socket** (p. 1900) is in Client or Server mode, true indicates that the mode is set to Client.*

  Returns

  *true if the **Socket** (p. 1900) is in Client mode, false otherwise.*

- virtual void **setNeedClientAuth** (bool value)

  *Sets the **Socket** (p. 1900) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
  This option only applies to sockets in the Server mode.
  If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

  Parameters

value	*The value indicating if a client is required to authenticate itself or not.*

- virtual bool **getNeedClientAuth** () **const**

  Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.
  This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

- virtual void **setWantClientAuth** (bool value)

  *Sets the **Socket** (p. 1900) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
  This option only applies to sockets in the Server mode.
  If the option is enabled an the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

  Parameters

value	*The value indicating if a client is requested to authenticate itself or not.*

- virtual bool **getWantClientAuth** () **const**

  Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that cleints that do not send a certificate are not considered to have failed authentication.
  This option is only useful when the socket is operating in server mode.

  Returns

  true if client authentication is required.

- int **read** (unsigned char ∗buffer, int size, int offset, int length)

  Reads the requested data from the Socket and write it into the passed in buffer.

- void **write** (**const** unsigned char ∗buffer, int size, int offset, int length)

  Writes the specified data in the passed in buffer to the Socket.

- int **available** ()

  Gets the number of bytes in the Socket buffer that can be read without blocking.

6.384.1 Detailed Description

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

Since

1.0

6.384.2 Constructor & Destructor Documentation

6.384.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters ∗ *parameters*)

6.384.2.2 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters ∗ *parameters,* const decaf::net::InetAddress ∗ *address,* int *port*)

6.384.2.3 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters ∗ *parameters,* const decaf::net::InetAddress ∗ *address,* int *port,* const decaf::net::InetAddress ∗ *localAddress,* int *localPort*)

6.384.2.4 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters ∗ *parameters,* const std::string & *host,* int *port*)

6.384.2.5 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (OpenSSLParameters ∗ *parameters,* const std::string & *host,* int *port,* const decaf::net::InetAddress ∗ *localAddress,* int *localPort*)

6.384.2.6 virtual decaf::internal::net::ssl::openssl::OpenSSLSocket::∼OpenSSLSocket ( ) [virtual]

6.384.3 Member Function Documentation

6.384.3.1 int decaf::internal::net::ssl::openssl::OpenSSLSocket::available ( )

Gets the number of bytes in the Socket buffer that can be read without blocking.

Returns

the number of bytes that can be read from the Socket without blocking.

Exceptions

IOException	if an I/O error occurs while performing this operation.

6.384.3.2 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::close ( ) [virtual]

Closes the **Socket** (p. 1900).

Once closed a **Socket** (p. 1900) cannot be connected or otherwise operated upon, a new **Socket** (p. 1900) instance must be created.

Exceptions

IOException	if an I/O error occurs while closing the **Socket** (p. 1900).

Reimplemented from **decaf::net::Socket** (p. 1905).

6.384.3.3 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::connect (const std::string & *host,* int *port,* int *timeout*) [virtual]

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 1932) is thrown. A timeout value of zero is treated as an inifinite timeout.

Parameters

host	The host name or IP address of the remote host to connect to.
port	The port on the remote host to connect to.
timeout	The number of Milliseconds to wait before treating the connection as failed.

Exceptions

IOException	Thrown if a failure occurred in the connect.
SocketTimeoutException *(p. 1932)*	if the timeout for connection is exceeded.
IllegalArguementException	if the timeout value is negative or the endpoint is invalid.

Reimplemented from **decaf::net::Socket** (p. 1905).

6.384.3.4 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledCipherSuites ( ) const [virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 1900).

Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLSocket** (p. 1950).

6.384.3.5 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledProtocols ( ) const [virtual]

Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 1900).

Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 1950).

6.384.3.6 virtual decaf::io::InputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getInputStream ( ) [virtual]

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 1900) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 1900).

Returns

The InputStream for this socket.

Exceptions

IOException	if an error occurs during creation of the InputStream, also if the **Socket** (p. 1900) is not connected or the input has been shutdown previously.

Reimplemented from **decaf::net::Socket** (p. 1906).

6.384.3.7 virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getNeedClientAuth ( ) const [virtual]

Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 1951).

6.384.3.8 virtual decaf::io::OutputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getOutputStream ( ) [virtual]

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 1900) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 1900) will also close the underlying **Socket** (p. 1900).

Returns

the OutputStream for this socket.

Exceptions

IOException	if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 1900) is closed or the output has been shutdown previously.

Reimplemented from **decaf::net::Socket** (p. 1907).

6.384.3.9 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedCipherSuites ( ) const [virtual]

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 1947).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 1900).

Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLSocket** (p. 1951).

6.384.3.10 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedProtocols ( ) const [virtual]

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 1947) instance.

Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 1951).

6.384.3.11 virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getUseClientMode ( ) const [virtual]

Gets whether this **Socket** (p. 1900) is in Client or Server mode, true indicates that the mode is set to Client.

Returns

true if the **Socket** (p. 1900) is in Client mode, false otherwise.

Implements **decaf::net::ssl::SSLSocket** (p. 1951).

6.384.3.12 virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getWantClientAuth ( ) const [virtual]

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that cleints that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 1952).

6.384.3.13 int decaf::internal::net::ssl::openssl::OpenSSLSocket::read (unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Reads the requested data from the Socket and write it into the passed in buffer.

Parameters

buffer	The buffer to read into
size	The size of the specified buffer
offset	The offset into the buffer where reading should start filling.
length	The number of bytes past offset to fill with data.

Returns

the actual number of bytes read or -1 if at EOF.

Exceptions

IOException	if an I/O error occurs during the read.
NullPointerException	if buffer is Null.
IndexOutOfBounds-Exception	if offset + length is greater than buffer size.

6.384.3.14 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::sendUrgentData (int *data*) [virtual]

Sends on byte of urgent data to the **Socket** (p. 1900).

Parameters

data	The value to write as urgent data, only the lower eight bits are sent.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 1910).

6.384.3.15 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledCipherSuites (const std::vector< std::string > & *suites*) [virtual]

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 1900) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites	An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLSocket** (p. 1952).

6.384.3.16 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledProtocols (const std::vector< std::string > & *protocols*) [virtual]

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 1900) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols	An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLSocket** (p. 1952).

6.384.3.17 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setNeedClientAuth (bool *value*) [virtual]

Sets the **Socket** (p. 1900) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

Parameters

value	The value indicating if a client is required to authenticate itself or not.

Implements **decaf::net::ssl::SSLSocket** (p. 1952).

6.384.3.18 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setOOBInline (bool *value*) [virtual]

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the **Socket** (p. 1900)'s InputStream, no notification is give.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException *(p. 1915)*	if an error is encountered while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 1910).

6.384.3.19 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setUseClientMode (bool *value*) [virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 1900), once a handshake has be initiated this socket remains the the set mode; client or server, for the life of this object.

Parameters

value	The mode setting, true for client or false for server.

Exceptions

IllegalArguementException	if the handshake process has begun and mode is lcoked.

Implements **decaf::net::ssl::SSLSocket** (p. 1953).

6.384.3.20 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setWantClientAuth (bool *value*) [virtual]

Sets the **Socket** (p. 1900) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

Parameters

value	The value indicating if a client is requested to authenticate itself or not.

Implements **decaf::net::ssl::SSLSocket** (p. 1953).

6.384.3.21 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownInput ( ) [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 1913).

6.384.3.22 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownOutput ( ) [virtual]

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OuputStream::write will throw an IOException.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 1913).

6.384.3.23 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::startHandshake ( ) [virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not require to support multiple handshakes and can throw an IOException to indicate an error.

Exceptions

IOException	if an I/O error occurs while performing the Handshake

Implements **decaf::net::ssl::SSLSocket** (p. 1954).

6.384.3.24 void decaf::internal::net::ssl::openssl::OpenSSLSocket::write (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Writes the specified data in the passed in buffer to the Socket.

Parameters

buffer	The buffer to write to the socket.
size	The size of the specified buffer.
offset	The offset into the buffer where the data to write starts at.
length	The number of bytes past offset to write.

Exceptions

IOException	if an I/O error occurs during the write.
NullPointerException	if buffer is Null.
IndexOutOfBounds-Exception	if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocket.h**

6.385 decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketException:

Public Member Functions

- **OpenSSLSocketException** () throw ()

  *Creates an new **OpenSSLSocketException** (p. 1571) with default values.*
- **OpenSSLSocketException** (**const Exception** &ex) throw ()

  Conversion Constructor from some other Exception.
- **OpenSSLSocketException** (**const OpenSSLSocketException** &ex) throw ()

Copy Constructor.

- **OpenSSLSocketException** (**const** char ∗file, **const** int lineNumber, **const** std::exception ∗**cause**, **const** char ∗msg,...) throw ()

  *Create a new **OpenSSLSocketException** (p. 1571) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (**const** std::exception ∗**cause**) throw ()

  *Creates a new **OpenSSLSocketException** (p. 1571) with the passed exception set as the cause of this exception.*
- **OpenSSLSocketException** (**const** char ∗file, **const** int lineNumber, **const** char ∗msg,...) throw ()

  *Create a new **OpenSSLSocketException** (p. 1571) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (**const** char ∗file, **const** int lineNumber) throw ()

  *Create a new **OpenSSLSocketException** (p. 1571) and initializes the file name and line number where this message occurred.*
- virtual **OpenSSLSocketException** ∗ **clone** () **const**

  Clones this exception.
- virtual ∼**OpenSSLSocketException** () throw ()

Protected Member Functions

- std::string **getErrorString** () **const**

  Gets and formats an error message string from the OpenSSL error stack.

6.385.1 Detailed Description

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

Since

1.0

6.385.2 Constructor & Destructor Documentation

6.385.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( ) throw ()

Creates an new **OpenSSLSocketException** (p. 1571) with default values.

6.385.2.2 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const Exception & *ex*) throw ()

Conversion Constructor from some other Exception.

Parameters

ex	An Exception object that should become this type of Exception.

6.385.2.3 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const OpenSSLSocketException & *ex*) throw ()

Copy Constructor.

Parameters

ex	The **OpenSSLSocketException** (p. 1571) whose values should be copied to this instance.

6.385.2.4 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()

Create a new **OpenSSLSocketException** (p. 1571) and 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 (can be null).
msg	The error message to report.
...	The list of primitives that are formatted into the message.

6.385.2.5 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const std::exception * *cause*) throw ()

Creates a new **OpenSSLSocketException** (p. 1571) with the passed exception set as the cause of this exception.

Parameters

cause	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.385.2.6 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()

Create a new **OpenSSLSocketException** (p. 1571) and 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 error message to report.
...	The list of primitives that are formatted into the message

6.385.2.7 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char * *file*, const int *lineNumber*) throw ()

Create a new **OpenSSLSocketException** (p. 1571) and initializes the file name and line number where this message occurred.

Sets the message to report by getting the complete set of error messages from the OpenSSL error stack and concatenating them into one string.

Parameters

file	The file name where exception occurs.
lineNumber	The line number where the exception occurred.

6.385.2.8 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketException::~OpenSSLSocketException ( ) throw () [virtual]

6.385.3 Member Function Documentation

6.385.3.1 virtual OpenSSLSocketException* decaf::internal::net::ssl::openssl::OpenSSLSocketException::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 this method.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 1916).

6.385.3.2 std::string decaf::internal::net::ssl::openssl::OpenSSLSocketException::getErrorString ( ) const [protected]

Gets and formats an error message string from the OpenSSL error stack.

Returns

a string containing the complete OpenSSL error string.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketException.h**

6.386 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketFactory:

Public Member Functions

- **OpenSSLSocketFactory** (**OpenSSLContextSpi** *parent)
- virtual ~**OpenSSLSocketFactory** ()
- virtual **decaf::net::Socket** * **createSocket** ()

  *Creates an unconnected **Socket** (p. 1900) object.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

Exceptions

IOException	*if the **Socket** (p. 1900) cannot be created.*

- virtual **decaf::net::Socket** * **createSocket** (**const decaf::net::InetAddress** *host, int port)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host to connect the socket to.*
port	*The port on the remote host to connect to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** * **createSocket** (**const decaf::net::InetAddress** *host, int port, **const decaf::net::InetAddress** *ifAddress, int localPort)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*
  *The **Socket** (p. 1900) will be bound to the specified local address and port.*

  Parameters

host	*The host to connect the socket to.*
port	*The port on the remote host to connect to.*
ifAddress	*The address on the local machine to bind the **Socket** (p. 1900) to.*
localPort	*The local port to bind the **Socket** (p. 1900) to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** * **createSocket** (**const** std::string &hostname, int port)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host name or IP address to connect the socket to.*
port	*The port on the remote host to connect to.*

  Returns

  *a new **Socket** (p. 1900) object, caller must free this object when done.*

  Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual **decaf::net::Socket** * **createSocket** (**const** std::string &name, int port, **const decaf::net::InetAddress** *ifAddress, int localPort)

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

  Parameters

host	*The host name or IP address to connect the socket to.*
port	*The port on the remote host to connect to.*
ifAddress	*The address on the local machine to bind the **Socket** (p. 1900) to.*

localPort	*The local port to bind the **Socket** (p. 1900) to.*

Returns

*a new **Socket** (p. 1900) object, caller must free this object when done.*

Exceptions

IOException	*if an I/O error occurs while creating the **Socket** (p. 1900) object.*
UnknownHostException (p. 2181)	*if the host name is not known.*

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

  Returns the list of cipher suites which are enabled by default.
  Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

  Returns

  an STL vector containing the list of cipher suites enabled by default.

  See also

  ***getSupportedCipherSuites()** (p. 1956)*

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

  Returns the names of the cipher suites which could be enabled for use on an SSL connection.
  Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

  Returns

  an STL vector containing the list of supported cipher suites.

  See also

  ***getDefaultCipherSuites()** (p. 1956)*

- virtual **decaf::net::Socket** * **createSocket** (**decaf::net::Socket** *socket, std::string host, int port, bool autoClose)

  Returns a socket layered over an existing socket connected to the named host, at the given port.
  This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

  Parameters

socket	*The existing socket to layer over.*
host	*The server host the original **Socket** (p. 1900) is connected to.*
port	*The server port the original **Socket** (p. 1900) is connected to.*
autoClose	*Should the layered over **Socket** (p. 1900) be closed when the topmost socket is closed.*

  Returns

  *a new **Socket** (p. 1900) instance that wraps the given **Socket** (p. 1900).*

  Exceptions

IOException	*if an I/O exception occurs while performing this operation.*
UnknownHostException (p. 2181)	*if the host is unknown.*

6.386.1 Detailed Description

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

Since

1.0

6.386.2 Constructor & Destructor Documentation

6.386.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::OpenSSLSocketFactory (OpenSSLContextSpi * *parent*)

6.386.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::~OpenSSLSocketFactory ( ) [virtual]

6.386.3 Member Function Documentation

6.386.3.1 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket ( ) [virtual]

Creates an unconnected **Socket** (p. 1900) object.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if the **Socket** (p. 1900) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 1917).

6.386.3.2 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const decaf::net::InetAddress * *host,* int *port*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1918).

6.386.3.3 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const decaf::net::InetAddress * *host,* int *port,* const decaf::net::InetAddress * *ifAddress,* int *localPort*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

The **Socket** (p. 1900) will be bound to the specified local address and port.

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1918).

6.386.3.4 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const std::string & *hostname,* int *port*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1919).

6.386.3.5 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const std::string & *name,* int *port,* const decaf::net::InetAddress * *ifAddress,* int *localPort*) [virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 1919).

6.386.3.6 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (decaf::net::Socket * *socket*, std::string *host*, int *port*, bool *autoClose*) [virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket	The existing socket to layer over.
host	The server host the original **Socket** (p. 1900) is connected to.
port	The server port the original **Socket** (p. 1900) is connected to.
autoClose	Should the layered over **Socket** (p. 1900) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 1900) instance that wraps the given **Socket** (p. 1900).

Exceptions

IOException	if an I/O exception occurs while performing this operation.
UnknownHostException *(p. 2181)*	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 1955).

6.386.3.7 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getDefaultCipherSuites ( ) [virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 1956)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 1956).

6.386.3.8 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getSupportedCipherSuites ( ) [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 1956)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 1956).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketFactory.h**

6.387 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference

An output stream for reading data from an OpenSSL Socket instance.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream:

Public Member Functions

- **OpenSSLSocketInputStream** (**OpenSSLSocket** *socket)
- virtual ~**OpenSSLSocketInputStream** ()
- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual void **close** ()

  Close - does nothing.

- virtual long long **skip** (long long num)

  Not supported.

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char *buffer, int size, int offset, int length)

6.387.1 Detailed Description

An output stream for reading data from an OpenSSL Socket instance.

Since

1.0

6.387.2 Constructor & Destructor Documentation

6.387.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::OpenSSLSocketInputStream (OpenSSLSocket * *socket*)

6.387.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::~OpenSSLSocketInputStream ( ) [virtual]

6.387.3 Member Function Documentation

6.387.3.1 virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.387.3.2 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::close ( ) [virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs while closing the **InputStream** (p. 1134).

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.387.3.3 virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadArrayBounded (unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.387.3.4 virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadByte ( ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1137).

6.387.3.5 virtual long long decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::skip (long long *num*) [virtual]

Not supported.

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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1140).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketInputStream.h**

6.388 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 1560) instance.

`#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.-h>`

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream:

Public Member Functions

- **OpenSSLSocketOutputStream** (**OpenSSLSocket** ∗socket)
- virtual ∼**OpenSSLSocketOutputStream** ()
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
  The object is generally no longer usable after calling close.

  Exceptions

IOException (p. 1198)	*if an error occurs while closing.*

  The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArrayBounded** (**const** unsigned char ∗buffer, int size, int offset, int length)

6.388.1 Detailed Description

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 1560) instance.

Since

1.0

6.388.2 Constructor & Destructor Documentation

6.388.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::OpenSSLSocketOutputStream (OpenSSLSocket ∗ *socket*)

6.388.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::∼OpenSSLSocketOutputStream ( ) [virtual]

6.388.3 Member Function Documentation

6.388.3.1 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(p. 1198)*	if an error occurs while closing.

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.388.3.2 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteArrayBounded (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.388.3.3 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteByte (unsigned char *c*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 1602).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketOutputStream.h**

6.389 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. 1584) object.*
- virtual ~**OpenWireFormat** ()
- virtual bool **hasNegotiator** () **const**

  *Returns true if this **WireFormat** (p. 2236) has a Negotiator that needs to wrap the Transport that uses it.*

  Returns

  *true if the **WireFormat** (p. 2236) provides a Negotiator.*
- virtual **Pointer** < **transport::Transport** > **createNegotiator** (**const Pointer**< **transport::Transport** > &transport)

  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. 2251) as a **Pointer**<**Transport**> (p. 1614).*

  Exceptions

UnsupportedOperation-Exception	*if the **WireFormat** (p. 2236) doesn't have a Negotiator.*

- void **addMarshaller** (**marshal::DataStreamMarshaller** *marshaler)

  Allows an external source to add marshalers to this object for types that may be marshaled or unmarhsaled.
- virtual void **marshal** (**const Pointer**< **commands::Command** > &command, **const activemq::transport-::Transport** *transport, **decaf::io::DataOutputStream** *out)

  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	*if an I/O error occurs.*

- virtual **Pointer**
  < **commands::Command** > **unmarshal** (**const activemq::transport::Transport** *transport, **decaf::io::-DataInputStream** *in)

  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	*if an I/O error occurs.*

- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** *object, **utils::BooleanStream** *bs)

  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)

  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)

  Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.
- **commands::DataStructure** * **looseUnmarshalNestedObject** (**decaf::io::DataInputStream** *dis)

  Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.
- void **looseMarshalNestedObject** (**commands::DataStructure** *o, **decaf::io::DataOutputStream** *data-Out)

  Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.
- void **renegotiateWireFormat** (**const commands::WireFormatInfo** &info)

  Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.
- void **setPreferedWireFormatInfo** (**const Pointer**< **commands::WireFormatInfo** > &info)

  Configures this object using the provided WireformatInfo object.
- **const Pointer**
  < **commands::WireFormatInfo** > & **getPreferedWireFormatInfo** () **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)

  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)

  Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

- void **destroyMarshalers** ()

  Cleans up all registered Marshallers and empties the dataMarshallers vector.

Static Protected Attributes

- static **const** unsigned char **NULL_TYPE**
- static **const** int **DEFAULT_VERSION**
- static **const** int **MAX_SUPPORTED_VERSION**

6.389.1 Constructor & Destructor Documentation

6.389.1.1 activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat (const decaf::util::Properties & *properties*)

Constructs a new **OpenWireFormat** (p. 1584) object.

Parameters

properties	- can contain optional config params.

6.389.1.2 virtual activemq::wireformat::openwire::OpenWireFormat::∼OpenWireFormat ( ) [virtual]

6.389.2 Member Function Documentation

6.389.2.1 void activemq::wireformat::openwire::OpenWireFormat::addMarshaller (marshal::DataStreamMarshaller * *marshaler*)

Allows an external source to add marshalers to this object for types that may be marshaled or unmarhsaled.

Parameters

marshaler	- the Marshaler to add to the collection.

6.389.2.2 virtual Pointer<transport::Transport> activemq::wireformat::openwire::OpenWireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) [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. 2251) as a **Pointer**<**Transport**> (p. 1614).

Exceptions

UnsupportedOperationException	if the **WireFormat** (p. 2236) doesn't have a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2237).

6.389.2.3 void activemq::wireformat::openwire::OpenWireFormat::destroyMarshalers ( ) [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

6.389.2.4 commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::doUnmarshal (decaf::io::DataInputStream * *dis*) [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	The DataInputStream to read from.

Returns

new DataStructure* that the caller owns.

Exceptions

IOException	if an error occurs during the unmarshal.

6.389.2.5 int activemq::wireformat::openwire::OpenWireFormat::getCacheSize ( ) const [inline]

Returns the currently set Cache size.

Returns

the current value of the broker's cache size.

6.389.2.6 long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration ( ) const [inline]

Gets the MaxInactivityDuration setting.

Returns

maximum inactivity duration value in milliseconds.

6.389.2.7 long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay ( ) const [inline]

Gets the MaxInactivityDurationInitialDelay setting.

Returns

maximum inactivity duration initial delay value in milliseconds.

6.389.2.8 const Pointer<commands::WireFormatInfo>& activemq::wireformat::openwire::OpenWireFormat::getPreferedWireFormatInfo ( ) const [inline]

Gets the Preferred WireFormatInfo object that this class holds.

Returns

pointer to a preferred WireFormatInfo object

6.389.2.9 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. 2237).

6.389.2.10 virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator ( ) const [inline, virtual]

Returns true if this **WireFormat** (p. 2236) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 2236) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2237).

6.389.2.11 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. 2238).

6.389.2.12 bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled ( ) const [inline]

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.389.2.13 bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled ( ) const [inline]

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.389.2.14 bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled ( ) const [inline]

Checks if the stackTraceEnabled flag is on.

Returns

true if the flag is on.

6.389.2.15 bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled ( ) const [inline]

Checks if the tcpNoDelayEnabled flag is on.

Returns

true if the flag is on.

6.389.2.16 bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled ( ) const [inline]

Checks if the tightEncodingEnabled flag is on.

Returns

true if the flag is on.

6.389.2.17 void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject (commands::DataStructure * *o*, decaf::io::DataOutputStream * *dataOut*)

Utility method to loosely Marshal an object that is derived from the DataStrucutre 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.

6.389.2.18 commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject (decaf::io::DataInputStream * *dis*)

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.

6.389.2.19 virtual void activemq::wireformat::openwire::OpenWireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport * *transport*, decaf::io::DataOutputStream * *out*) [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	if an I/O error occurs.

Implements **activemq::wireformat::WireFormat** (p. 2238).

6.389.2.20 void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat (const commands::WireFormatInfo & *info*)

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.

6.389.2.21 void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled (bool *cacheEnabled*) [inline]

Sets if the cacheEnabled flag is on.

Parameters

cacheEnabled	- true to turn flag is on

6.389.2.22 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.

6.389.2.23 void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration (long long *value*) [inline]

Sets the MaxInactivityDuration setting.

Parameters

value	- the Max inactivity duration value in milliseconds.

6.389.2.24 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.

6.389.2.25 void activemq::wireformat::openwire::OpenWireFormat::setPreferedWireFormatInfo (const Pointer< commands::WireFormatInfo > & *info*)

Configures this object using the provided WireformatInfo object.

Parameters

info	A WireFormatInfo object, takes ownership.

Exceptions

IllegalStateException	if the **WireFormat** (p. 2236) object has not been initialized.

6.389.2.26 void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled (bool *sizePrefixDisabled*) [inline]

Sets if the sizePrefixDisabled flag is on.

Parameters

sizePrefix-Disabled	- true to turn flag is on

6.389.2.27 void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled (bool *stackTraceEnabled*) [inline]

Sets if the stackTraceEnabled flag is on.

Parameters

stackTrace-Enabled	- true to turn flag is on

6.389.2.28 void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*) [inline]

Sets if the tcpNoDelayEnabled flag is on.

Parameters

tcpNoDelay-Enabled	- true to turn flag is on

6.389.2.29 void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled (bool *tightEncodingEnabled*) [inline]

Sets if the tightEncodingEnabled flag is on.

Parameters

tightEncoding-Enabled	- true to turn flag is on

6.389.2.30 void activemq::wireformat::openwire::OpenWireFormat::setVersion (int *version*) [virtual]

Set the current Wireformat Version.

Parameters

version	An int that identifies the version

Exceptions

IllegalArgumentException	if the version given is not supported.

Implements **activemq::wireformat::WireFormat** (p. 2238).

6.389.2.31 virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1 (commands::DataStructure * *object,* utils::BooleanStream * *bs*) [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.

6.389.2.32 void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2 (commands::DataStructure * *o,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*)

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	- DataOuputStream for writing
bs	- BooleanStream

Exceptions

IOException	if an error occurs.

6.389.2.33 commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject (decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*)

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.

6.389.2.34 virtual Pointer<commands::Command> activemq::wireformat::openwire::OpenWireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) [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	if an I/O error occurs.

Implements **activemq::wireformat::WireFormat** (p. 2238).

6.389.3 Field Documentation

6.389.3.1 const int activemq::wireformat::openwire::OpenWireFormat::DEFAULT_VERSION [static, protected]

6.389.3.2 const int activemq::wireformat::openwire::OpenWireFormat::MAX_SUPPORTED_VERSION [static, protected]

6.389.3.3 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**

6.390 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)

  *Creates a new **WireFormat** (p. 2236) Object passing it a set of properties from which it can obtain any optional settings.*

6.390.1 Constructor & Destructor Documentation

6.390.1.1 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

6.390.1.2 virtual activemq::wireformat::openwire::OpenWireFormatFactory::~OpenWireFormatFactory ( ) [inline, virtual]

6.390.2 Member Function Documentation

6.390.2.1 virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) [virtual]

Creates a new **WireFormat** (p. 2236) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties	The Properties for this **WireFormat** (p. 2236).

Returns

Pointer to a new instance of a **WireFormat** (p. 2236) object.

Exceptions

IllegalStateException	if the factory has not been initialized.

Implements **activemq::wireformat::WireFormatFactory** (p. 2239).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatFactory.h**

6.391 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.

6.391.1 Constructor & Destructor Documentation

6.391.1.1 activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator (OpenWireFormat * *wireFormat,* const Pointer< transport::Transport > & *next*)

Constructor - Initializes this object around another Transport.

Parameters

wireFormat	- The **WireFormat** (p. 2236) object we use to negotiate
next	- The next transport in the chain

6.391.1.2 virtual activemq::wireformat::openwire::OpenWireFormatNegotiator::~OpenWireFormatNegotiator ( ) [virtual]

6.391.2 Member Function Documentation

6.391.2.1 virtual void activemq::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. 2170).

6.391.2.2 virtual void activemq::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. 2173).

6.391.2.3 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::oneway (const Pointer< commands::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. 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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2173).

6.391.2.4 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.

6.391.2.5 virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWire-FormatNegotiator::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. 2174).

6.391.2.6 virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormat-Negotiator::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. 2174).

6.391.2.7 virtual void activemq::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. 2175).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatNegotiator.h**

6.392 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

Used to allow a MockTransport to generate response commands to OpenWire Commands.

`#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::LinkedList**< **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.

6.392.1 Detailed Description

Used to allow a MockTransport to generate response commands to OpenWire Commands.

6.392.2 Constructor & Destructor Documentation

6.392.2.1 activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder ( ) [inline]

6.392.2.2 virtual activemq::wireformat::openwire::OpenWireResponseBuilder::~OpenWireResponseBuilder ( ) [inline, virtual]

6.392.3 Member Function Documentation

6.392.3.1 virtual void activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncoming-Commands (const Pointer< commands::Command > & *command*, decaf::util::LinkedList< 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. 1785).

6.392.3.2 virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWire-ResponseBuilder::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. 1785).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireResponseBuilder.h**

6.393 decaf::io::OutputStream Class Reference

Base interface for any class that wants to represent an output stream of bytes.

`#include <src/main/decaf/io/OutputStream.h>`

Inheritance diagram for decaf::io::OutputStream:

Public Member Functions

- **OutputStream** ()

- virtual ~**OutputStream** ()
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
  The object is generally no longer usable after calling close.

  Exceptions

IOException (p. 1198)	*if an error occurs while closing.*

- virtual void **flush** ()

  Flushes this stream by writing any buffered output to the underlying stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual void **write** (unsigned char c)

  Writes a single byte to the output stream.
- virtual void **write** (**const** unsigned char *buffer, int size)

  Writes an array of bytes to the output stream.
- virtual void **write** (**const** unsigned char *buffer, int size, int offset, int length)

  Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.
- virtual std::string **toString** () **const**

  Output a String representation of this object.
- virtual void **lock** ()

  Locks the object.
- virtual bool **tryLock** ()

  Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** ()

  Unlocks the object.
- virtual void **wait** ()

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()

  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()

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

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)=0
- virtual void **doWriteArray** (**const** unsigned char *buffer, int size)
- virtual void **doWriteArrayBounded** (**const** unsigned char *buffer, int size, int offset, int length)

6.393.1 Detailed Description

Base interface for any class that wants to represent an output stream of bytes.

Since

1.0

6.393.2 Constructor & Destructor Documentation

6.393.2.1 decaf::io::OutputStream::OutputStream ( )

6.393.2.2 virtual decaf::io::OutputStream::~OutputStream ( ) [virtual]

6.393.3 Member Function Documentation

6.393.3.1 virtual void decaf::io::OutputStream::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(*p. *1198)*	if an error occurs while closing.

The default implementation of this method does nothing.

Implements **decaf::io::Closeable** (p. 633).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 923), **decaf::io::FilterOutputStream** (p. 1039), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 2086), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 1583), **decaf::internal::io::StandardErrorOutputStream** (p. 1960), and **decaf::internal::io::StandardOutputStream** (p. 1962).

6.393.3.2 virtual void decaf::io::OutputStream::doWriteArray (const unsigned char * *buffer,* int *size*) [protected, virtual]

Reimplemented in **decaf::io::FilterOutputStream** (p. 1039), and **decaf::io::BufferedOutputStream** (p. 462).

6.393.3.3 virtual void decaf::io::OutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 924), **decaf::io::DataOutputStream** (p. 862), **decaf::io::ByteArrayOutputStream** (p. 534), **decaf::io::FilterOutputStream** (p. 1039), **decaf::io::BufferedOutputStream** (p. 462), **decaf::util::zip::CheckedOutputStream** (p. 628), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 2086), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 1584), **activemq::io::LoggingOutputStream** (p. 1324), **decaf::internal::io::StandardErrorOutputStream** (p. 1960), and **decaf::internal::io::StandardOutputStream** (p. 1963).

6.393.3.4 virtual void decaf::io::OutputStream::doWriteByte (unsigned char *value*) [protected, pure virtual]

Implemented in **decaf::util::zip::DeflaterOutputStream** (p. 924), **decaf::io::DataOutputStream** (p. 862), **decaf::io::ByteArrayOutputStream** (p. 534), **decaf::io::FilterOutputStream** (p. 1039), **decaf::io::BufferedOutputStream** (p. 463), **decaf::util::zip::CheckedOutputStream** (p. 628), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 2086), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 1584), **activemq::io::LoggingOutputStream** (p. 1324), **decaf::internal::io::StandardErrorOutputStream** (p. 1960), and **decaf::internal::io::StandardOutputStream** (p. 1963).

6.393.3.5 virtual void decaf::io::OutputStream::flush ( ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

The default implementation of this method does nothing.

Implements **decaf::io::Flushable** (p. 1068).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 463), **decaf::io::FilterOutputStream** (p. 1039), **decaf::internal::io::StandardErrorOutputStream** (p. 1960), and **decaf::internal::io::StandardOutputStream** (p. 1963).

6.393.3.6 virtual void decaf::io::OutputStream::lock ( ) [inline, virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

6.393.3.7 virtual void decaf::io::OutputStream::notify ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2047).

6.393.3.8 virtual void decaf::io::OutputStream::notifyAll ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2048).

6.393.3.9 virtual std::string decaf::io::OutputStream::toString ( ) const [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

Returns

a string representation of the object.

Reimplemented in **decaf::io::ByteArrayOutputStream** (p. 535), and **decaf::io::FilterOutputStream** (p. 1040).

6.393.3.10 virtual bool decaf::io::OutputStream::tryLock ( ) [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. 2048).

6.393.3.11 virtual void decaf::io::OutputStream::unlock ( ) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

6.393.3.12 virtual void decaf::io::OutputStream::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.393.3.13 virtual void decaf::io::OutputStream::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.393.3.14 virtual void decaf::io::OutputStream::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

6.393.3.15 virtual void decaf::io::OutputStream::write (unsigned char *c*) [virtual]

Writes a single byte to the output stream.

The default implementation of this method calls the pure virtual method doWriteByte which must be implemented by any subclass of the **OutputStream** (p. 1600).

Parameters

c	The byte to write to the sink.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

6.393.3.16 virtual void decaf::io::OutputStream::write (const unsigned char * *buffer,* int *size*) [virtual]

Writes an array of bytes to the output stream.

The entire contents of the given vector are written to the output stream.

The default implementation of this method simply calls the doWriteArray which writes the contents of the array using the doWriteByte method repeatedly. It is recommended that a subclass override doWriteArray to provide

more performant means of writing the array.

Parameters

buffer	The vector of bytes to write.
size	The size of the buffer passed.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.
NullPointerException	thrown if buffer is Null.
IndexOutOfBounds-Exception	if size value is negative.

6.393.3.17 virtual void decaf::io::OutputStream::write (const unsigned char * *buffer*, int *size*, int *offset*, int *length*) [virtual]

Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.

The default implementation of this method simply calls the doWriteArrayBounded method which writes the contents of the array using the doWriteByte method repeatedly. It is recommended that a subclass override doWriteArray-Bounded to provide more performant means of writing the array.

Parameters

buffer	The array of bytes to write.
size	The size of the buffer array passed.
offset	The position to start writing in buffer.
length	The number of bytes from the buffer to be written.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.
NullPointerException	thrown if buffer is Null.
IndexOutOfBounds-Exception	if the offset + length $>$ size. or one of the parameters is negative.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**OutputStream.h**

6.394 decaf::io::OutputStreamWriter Class Reference

A class for turning a character stream into a byte stream.

`#include <src/main/decaf/io/OutputStreamWriter.h>`

Inheritance diagram for decaf::io::OutputStreamWriter:

Public Member Functions

- **OutputStreamWriter** (**OutputStream** *stream, bool own=false)

*Creates a new **OutputStreamWriter** (p. 1606).*

- virtual ~**OutputStreamWriter** ()
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- virtual void **flush** ()

  Flushes this stream by writing any buffered output to the underlying stream.

Protected Member Functions

- virtual void **doWriteArrayBounded** (**const** char *buffer, int size, int offset, int length)

  Override this method to customize the functionality of the method write(char buffer, int size, int offset, int length).*
- virtual void **checkClosed** () **const**

6.394.1 Detailed Description

A class for turning a character stream into a byte stream.

See also

InputStreamReader (p. 1142)

Since

1.0

6.394.2 Constructor & Destructor Documentation

6.394.2.1 decaf::io::OutputStreamWriter::OutputStreamWriter (OutputStream * *stream*, bool *own* = false)

Creates a new **OutputStreamWriter** (p. 1606).

Parameters

stream	The **OutputStream** (p. 1600) to wrap. (cannot be NULL).
own	Indicates whether this instance own the given **OutputStream** (p. 1600). If true then the **OutputStream** (p. 1600) is destroyed when this class is.

Exceptions

NullPointerException	if the stream is NULL.

6.394.2.2 virtual decaf::io::OutputStreamWriter::~OutputStreamWriter ( ) [virtual]

6.394.3 Member Function Documentation

6.394.3.1 virtual void decaf::io::OutputStreamWriter::checkClosed ( ) const [protected, virtual]

6.394.3.2 virtual void decaf::io::OutputStreamWriter::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(p. 1198)*	if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 633).

6.394.3.3 virtual void decaf::io::OutputStreamWriter::doWriteArrayBounded (const char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Override this method to customize the functionality of the method write(char* buffer, int size, int offset, int length).

All subclasses must override this method to provide the basic **Writer** (p. 2255) functionality.

Implements **decaf::io::Writer** (p. 2257).

6.394.3.4 virtual void decaf::io::OutputStreamWriter::flush ( ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Implements **decaf::io::Flushable** (p. 1068).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**OutputStreamWriter.h**

6.395 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 ***DataStructure*** *(p. 877) Type as defined in CommandTypes.h.*
- virtual **PartialCommand** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  Compares the ***DataStructure*** *(p. 877) 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 Attributes

- int **commandId**
- std::vector< unsigned char > **data**

6.395.1 Constructor & Destructor Documentation

6.395.1.1 activemq::commands::PartialCommand::PartialCommand ( )

6.395.1.2 virtual activemq::commands::PartialCommand::~PartialCommand ( ) [virtual]

6.395.2 Member Function Documentation

6.395.2.1 virtual PartialCommand* activemq::commands::PartialCommand::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1246).

6.395.2.2 virtual void activemq::commands::PartialCommand::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. 879).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1246).

6.395.2.3 virtual bool activemq::commands::PartialCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1246).

6.395.2.4 virtual int activemq::commands::PartialCommand::getCommandId ( ) const [virtual]

6.395.2.5 virtual const std::vector<unsigned char>& activemq::commands::PartialCommand::getData ( ) const [virtual]

6.395.2.6 virtual std::vector<unsigned char>& activemq::commands::PartialCommand::getData ( ) [virtual]

6.395.2.7 virtual unsigned char activemq::commands::PartialCommand::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1246).

6.395.2.8 virtual void activemq::commands::PartialCommand::setCommandId (int *commandId*) [virtual]

6.395.2.9 virtual void activemq::commands::PartialCommand::setData (const std::vector< unsigned char > & *data*) [virtual]

6.395.2.10 virtual std::string activemq::commands::PartialCommand::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1247).

6.395.3 Field Documentation

6.395.3.1 int activemq::commands::PartialCommand::commandId [protected]

6.395.3.2 std::vector<unsigned char> activemq::commands::PartialCommand::data [protected]

6.395.3.3 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**

6.396 activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 1611).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/PartialCommandMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual ∼**PartialCommandMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.396.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 1611).

NOTE!: This file is 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::generated::PartialCommandMarshaller::PartialCommandMarshaller ( ) [inline]

6.396.2.2 virtual activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::∼PartialCommandMarshaller ( ) [inline, virtual]

6.396.3 Member Function Documentation

6.396.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1248).

6.396.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1248).

6.396.3.3 virtual void activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1248).

6.396.3.4 virtual void activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1249).

6.396.3.5 virtual int activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1249).

6.396.3.6 virtual void activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1249).

6.396.3.7 virtual void activemq::wireformat::openwire::marshal::generated::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LastPartialCommandMarshaller** (p. 1250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**PartialCommandMarshaller.h**

6.397 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 1614) that is a template on a Type and is **Thread** (p. 2094) 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. 1614) 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** &)

Dynamic Cast constructor.

- virtual ~**Pointer** () throw ()
- void **reset** (T *value)

  *Resets the **Pointer** (p. 1614) to hold the new value.*
- T * **release** ()

  *Releases the **Pointer** (p. 1614) held and resets the internal pointer value to Null.*
- **PointerType get** () **const**

  *Gets the real pointer that is contained within this **Pointer** (p. 1614).*
- void **swap** (**Pointer** &value) throw ()

  ***Exception** (p. 990) Safe Swap Function.*
- **Pointer** & **operator=** (**const Pointer** &right) throw ()

  *Assigns the value of right to this **Pointer** (p. 1614) 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)

6.397.1 Detailed Description

template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter>class decaf::lang::Pointer< T, REFCOUNTER >

Decaf's implementation of a Smart **Pointer** (p. 1614) that is a template on a Type and is **Thread** (p. 2094) Safe if the default Reference Counter is used.

This **Pointer** (p. 1614) 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. 1614) 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. 1614) in a STL container that requires it, **Pointer** (p. 1614) provides an implementation of std::less.

Since

1.0

6.397.2 Member Typedef Documentation

6.397.2.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER decaf::lang::Pointer< T, REFCOUNTER >::CounterType

6.397.2.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T* decaf::lang::Pointer< T, REFCOUNTER >::PointerType

6.397.2.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T& decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType

6.397.3 Constructor & Destructor Documentation

6.397.3.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> 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().

6.397.3.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const PointerType *value*) [inline, explicit]

Explicit Constructor, creates a **Pointer** (p. 1614) 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.

6.397.3.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> 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.

6.397.3.4 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> 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.

6.397.3.5 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> 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. 1614) object.

Parameters

value	- **Pointer** (p. 1614) instance to cast to this type.

6.397.3.6 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & *value,* const DYNAMIC_CAST_TOKEN &) [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. 1614) object. If the cast fails and return NULL then this method throws a ClassCastException.

Parameters

value	- **Pointer** (p. 1614) instance to cast to this type.

Exceptions

ClassCastException	if the dynamic cast returns NULL

6.397.3.7 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> virtual decaf::lang::Pointer< T, REFCOUNTER >::~Pointer ( ) throw () [inline, virtual]

6.397.4 Member Function Documentation

6.397.4.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::dynamicCast ( ) const [inline]

6.397.4.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::get ( ) const [inline]

Gets the real pointer that is contained within this **Pointer** (p. 1614).

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. 1614). Use at your own risk.

Returns

the contained pointer.

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals(), activemq::state::ConnectionState::getTransactionState(), decaf::lang::Pointer< TransactionId >::operator!=(), decaf::lang::operator!=(), std::less< decaf::lang::Pointer< T > >::operator()(), decaf::lang::Pointer< TransactionId >::operator==(), decaf::lang::operator==(), decaf::util::concurrent::LinkedBlockingQueue< E >::peek(), and activemq::state::ConnectionState::removeTempDestination().

6.397.4.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool decaf::lang::Pointer< T, REFCOUNTER >::operator! ( ) const [inline]

6.397.4.4 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator!= (const Pointer< T1, R1 > & *right*) const [inline]

6.397.4.5 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> 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

6.397.4.6 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType decaf::lang::Pointer< T, REFCOUNTER >::operator* ( ) const [inline]

6.397.4.7 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> 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

6.397.4.8 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::operator-> ( ) const [inline]

6.397.4.9 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T, REFCOUNTER > & *right*) throw () [inline]

Assigns the value of right to this **Pointer** (p. 1614) and increments the reference Count.

Parameters

right	- **Pointer** (p. 1614) on the right hand side of an operator= call to this.

6.397.4.10 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 , typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T1, R1 > & *right*) throw () [inline]

6.397.4.11 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator== (const Pointer< T1, R1 > & *right*) const [inline]

6.397.4.12 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> T* decaf::lang::Pointer< T, REFCOUNTER >::release ( ) [inline]

Releases the **Pointer** (p. 1614) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 1614) 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. 1614) object, the pointer is no longer owned by this **Pointer** (p. 1614) and won't be freed when this **Pointer** (p. 1614) goes out of scope.

Referenced by decaf::lang::Pointer< TransactionId >::Pointer().

6.397.4.13 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::reset (T * *value*) [inline]

Resets the **Pointer** (p. 1614) 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. 1614) to a NULL pointer.

Parameters

value	- The new value to contain.

6.397.4.14 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::staticCast ( ) const [inline]

6.397.4.15 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::swap (Pointer< T, REFCOUNTER > & *value*) throw () [inline]

Exception (p. 990) Safe Swap Function.

Parameters

value	- the value to swap with this.

Referenced by decaf::lang::Pointer< TransactionId >::operator=(), and decaf::lang::Pointer< TransactionId >::swap().

6.397.5 Friends And Related Function Documentation

6.397.5.1 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const Pointer< T, REFCOUNTER > & *left,* const T * *right*) [friend]

6.397.5.2 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= (const T * *left,* const Pointer< T, REFCOUNTER > & *right*) [friend]

6.397.5.3 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== (const Pointer< T, REFCOUNTER > & *left,* const T * *right*) [friend]

6.397.5.4 template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> 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**

6.398 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. 1614) instance.

`#include <src/main/decaf/lang/Pointer.h>`

Inheritance diagram for decaf::lang::PointerComparator< T, R >:

Public Member Functions

- virtual ~**PointerComparator** ()
- 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**

6.398.1 Detailed Description

template<typename T, typename R = decaf::util::concurrent::atomic::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. 1614) 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. 1614) 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.

6.398.2 Constructor & Destructor Documentation

6.398.2.1 template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual decaf::lang::PointerComparator< T, R >::~PointerComparator ( ) [inline, virtual]

6.398.3 Member Function Documentation

6.398.3.1 template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual int decaf::lang::PointerComparator< T, R >::compare (const Pointer< T, R > & *left,* const Pointer< T, R > & *right*) const [inline, virtual]

6.398.3.2 template<typename T , typename R = decaf::util::concurrent::atomic::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**

6.399 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** () throw ()

  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** ()

  Returns this session back to the pool, but does not close or destroy the internal session object.
- virtual void **start** ()

  Starts the service.
- virtual void **stop** ()

  Stops this service.
- virtual void **commit** ()

  Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()

  Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** ()

  Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **cms::MessageConsumer** * **createConsumer** (**const cms::Destination** *destination)

  Creates a MessageConsumer for the specified destination.
- virtual **cms::MessageConsumer** * **createConsumer** (**const cms::Destination** *destination, **const** std::string &selector)

  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)

  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)

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)

  First checks the internal consumer cache and creates on if none exist for the given destination, selector, noLocal.

- virtual **cms::MessageProducer** ∗ **createProducer** (**const cms::Destination** ∗destination)

  Creates a MessageProducer to send messages to the specified destination.

- virtual **cms::MessageProducer** ∗ **createCachedProducer** (**const cms::Destination** ∗destination)

  First checks the internal producer cache and creates one if none exist for the given destination.

- virtual **cms::QueueBrowser** ∗ **createBrowser** (**const cms::Queue** ∗queue)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::QueueBrowser** ∗ **createBrowser** (**const cms::Queue** ∗queue, **const** std::string &selector)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::Queue** ∗ **createQueue** (**const** std::string &queueName)

  Creates a queue identity given a Queue name.

- virtual **cms::Topic** ∗ **createTopic** (**const** std::string &topicName)

  Creates a topic identity given a Queue name.

- virtual **cms::TemporaryQueue** ∗ **createTemporaryQueue** ()

  Creates a TemporaryQueue object.

- virtual **cms::TemporaryTopic** ∗ **createTemporaryTopic** ()

  Creates a TemporaryTopic object.

- virtual **cms::Message** ∗ **createMessage** ()

  Creates a new Message.

- virtual **cms::BytesMessage** ∗ **createBytesMessage** ()

  Creates a BytesMessage.

- virtual **cms::BytesMessage** ∗ **createBytesMessage** (**const** unsigned char ∗bytes, int bytesSize)

  Creates a BytesMessage and sets the payload to the passed value.

- virtual **cms::StreamMessage** ∗ **createStreamMessage** ()

  Creates a new StreamMessage.

- virtual **cms::TextMessage** ∗ **createTextMessage** ()

  Creates a new TextMessage.

- virtual **cms::TextMessage** ∗ **createTextMessage** (**const** std::string &text)

  Creates a new TextMessage and set the text to the value given.

- virtual **cms::MapMessage** ∗ **createMapMessage** ()

  Creates a new MapMessage.

- virtual **cms::Session::AcknowledgeMode getAcknowledgeMode** () **const**

  Returns the acknowledgment mode of the session.

- virtual bool **isTransacted** () **const**

  Gets if the Sessions is a Transacted Session.

- virtual void **unsubscribe** (**const** std::string &name)

  Unsubscribes a durable subscription that has been created by a client.

6.399.1 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.

6.399.2 Constructor & Destructor Documentation

6.399.2.1 activemq::cmsutil::PooledSession::PooledSession (SessionPool * *pool,* cms::Session * *session*)

6.399.2.2 virtual activemq::cmsutil::PooledSession::~PooledSession ( ) throw () [virtual]

Does nothing.

6.399.3 Member Function Documentation

6.399.3.1 virtual void activemq::cmsutil::PooledSession::close ( ) [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object.

Exceptions

CMSException	if an error occurs while performing this operation.

Implements **cms::Session** (p. 1833).

6.399.3.2 virtual void activemq::cmsutil::PooledSession::commit ( ) [inline, virtual]

Commits 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. 1834).

References cms::Session::commit().

6.399.3.3 virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * *queue*) [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.

Implements **cms::Session** (p. 1834).

6.399.3.4 virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * *queue,* const std::string & *selector*) [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

CMSException	- If an internal error occurs.
InvalidDestinationException	- if the destination given is invalid.

Implements **cms::Session** (p. 1834).

6.399.3.5 virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage ( ) [inline, virtual]

Creates a BytesMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1835).

References cms::Session::createBytesMessage().

6.399.3.6 virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage (const unsigned char * *bytes,* int *bytesSize*) [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

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1835).

References cms::Session::createBytesMessage().

6.399.3.7 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer (const cms::Destination * *destination,* const std::string & *selector,* bool *noLocal*) [virtual]

First checks the internal consumer cache and creates on 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::CMSException *(p. 640)*	if something goes wrong.

6.399.3.8 virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createCachedProducer (const cms::Destination * *destination*) [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::CMSException *(p. 640)*	if something goes wrong.

6.399.3.9 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * *destination*) [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. 1835).

References cms::Session::createConsumer().

6.399.3.10 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * *destination*, const std::string & *selector*) [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. 1836).

References cms::Session::createConsumer().

6.399.3.11 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * *destination*, const std::string & *selector*, bool *noLocal*) [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. 1836).

References cms::Session::createConsumer().

6.399.3.12 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createDurableConsumer (const cms::Topic * *destination*, const std::string & *name*, const std::string & *selector*, bool *noLocal* = false) [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. 1837).

References cms::Session::createDurableConsumer().

6.399.3.13 virtual cms::MapMessage* activemq::cmsutil::PooledSession::createMapMessage ( ) [inline, virtual]

Creates a new MapMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1837).

References cms::Session::createMapMessage().

6.399.3.14 virtual cms::Message* activemq::cmsutil::PooledSession::createMessage ( ) [inline, virtual]

Creates a new Message.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1837).

References cms::Session::createMessage().

6.399.3.15 virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer (const cms::Destination * *destination*) [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. 1838).

References cms::Session::createProducer().

6.399.3.16 virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue (const std::string & *queueName*) [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

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1838).

References cms::Session::createQueue().

6.399.3.17 virtual cms::StreamMessage* activemq::cmsutil::PooledSession::createStreamMessage ( ) [inline, virtual]

Creates a new StreamMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1838).

References cms::Session::createStreamMessage().

6.399.3.18 virtual cms::TemporaryQueue* activemq::cmsutil::PooledSession::createTemporaryQueue ( ) [inline, virtual]

Creates a TemporaryQueue object.

Returns

new TemporaryQueue pointer that is owned by the caller.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1839).

References cms::Session::createTemporaryQueue().

6.399.3.19 virtual cms::TemporaryTopic* activemq::cmsutil::PooledSession::createTemporaryTopic ( ) [inline, virtual]

Creates a TemporaryTopic object.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1839).

References cms::Session::createTemporaryTopic().

6.399.3.20 virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage ( ) [inline, virtual]

Creates a new TextMessage.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1839).

References cms::Session::createTextMessage().

6.399.3.21 virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage (const std::string & *text*) [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. 1839).

References cms::Session::createTextMessage().

6.399.3.22 virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic (const std::string & *topicName*) [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. 1840).

References cms::Session::createTopic().

6.399.3.23 virtual cms::Session::AcknowledgeMode activemq::cmsutil::PooledSession::getAcknowledgeMode ( ) const [inline, virtual]

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1840).

References cms::Session::getAcknowledgeMode().

6.399.3.24 virtual cms::Session* activemq::cmsutil::PooledSession::getSession ( ) [inline, virtual]

Returns a non-constant reference to the internal session object.

Returns

the session object.

6.399.3.25 virtual const cms::Session* activemq::cmsutil::PooledSession::getSession ( ) const [inline, virtual]

Returns a constant reference to the internal session object.

Returns

the session object.

6.399.3.26 virtual bool activemq::cmsutil::PooledSession::isTransacted ( ) const [inline, virtual]

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1840).

References cms::Session::isTransacted().

6.399.3.27 virtual void activemq::cmsutil::PooledSession::recover ( ) [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. 1840).

References cms::Session::recover().

6.399.3.28 virtual void activemq::cmsutil::PooledSession::rollback ( ) [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. 1841).

References cms::Session::rollback().

6.399.3.29 virtual void activemq::cmsutil::PooledSession::start ( ) [inline, virtual]

Starts the service.

Exceptions

CMSException	if an internal error occurs while starting.

Implements **cms::Startable** (p. 1964).

References cms::Startable::start().

6.399.3.30 virtual void activemq::cmsutil::PooledSession::stop ( ) [inline, virtual]

Stops this service.

Exceptions

CMSException	- if an internal error occurs while stopping the Service.

Implements **cms::Stoppable** (p. 2017).

References cms::Stoppable::stop().

6.399.3.31 virtual void activemq::cmsutil::PooledSession::unsubscribe (const std::string & *name*) [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

CMSException	- If an internal error occurs.

Implements **cms::Session** (p. 1841).

References cms::Session::unsubscribe().

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

6.400 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 ()

6.400.1 Constructor & Destructor Documentation

6.400.1.1 decaf::net::PortUnreachableException::PortUnreachableException ( ) throw () [inline]

Default Constructor.

6.400.1.2 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

6.400.1.3 decaf::net::PortUnreachableException::PortUnreachableException (const PortUnreachableException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.400.1.4 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

6.400.1.5 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.

6.400.1.6 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

6.400.1.7 virtual decaf::net::PortUnreachableException::~PortUnreachableException ( ) throw () [inline, virtual]

6.400.2 Member Function Documentation

6.400.2.1 virtual PortUnreachableException* decaf::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. 1916).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**PortUnreachableException.h**

6.401 activemq::core::PrefetchPolicy Class Reference

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

`#include <src/main/activemq/core/PrefetchPolicy.h>`

Inheritance diagram for activemq::core::PrefetchPolicy:

Public Member Functions

- virtual ~**PrefetchPolicy** ()
- virtual void **setDurableTopicPrefetch** (int value)=0

  Sets the amount of prefetched messages for a Durable Topic.
- virtual int **getDurableTopicPrefetch** () **const** =0

  Gets the amount of messages to prefetch for a Durable Topic.
- virtual void **setQueuePrefetch** (int value)=0

  Sets the amount of prefetched messages for a Queue.
- virtual int **getQueuePrefetch** () **const** =0

  Gets the amount of messages to prefetch for a Queue.
- virtual void **setQueueBrowserPrefetch** (int value)=0

  Sets the amount of prefetched messages for a Queue Browser.
- virtual int **getQueueBrowserPrefetch** () **const** =0

  Gets the amount of messages to prefetch for a Queue Browser.
- virtual void **setTopicPrefetch** (int value)=0

  Sets the amount of prefetched messages for a Topic.
- virtual int **getTopicPrefetch** () **const** =0

  Gets the amount of messages to prefetch for a Topic.
- virtual int **getMaxPrefetchLimit** (int value) **const** =0

  Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.
- virtual **PrefetchPolicy** * **clone** () **const** =0

  Clone the Policy and return a new pointer to that clone.
- virtual void **configure** (**const decaf::util::Properties** &properties)

  Checks the supplied properties object for properties matching the configurable settings of this class.

Protected Member Functions

- **PrefetchPolicy** ()

6.401.1 Detailed Description

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

Since

3.2.0

6.401.2 Constructor & Destructor Documentation

6.401.2.1 activemq::core::PrefetchPolicy::PrefetchPolicy ( ) [protected]

6.401.2.2 virtual activemq::core::PrefetchPolicy::~PrefetchPolicy ( ) [virtual]

6.401.3 Member Function Documentation

6.401.3.1 virtual PrefetchPolicy* activemq::core::PrefetchPolicy::clone ( ) const [pure virtual]

Clone the Policy and return a new pointer to that clone.

Returns

pointer to a new **PrefetchPolicy** (p. 1635) instance that is a clone of this one.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 888).

6.401.3.2 virtual void activemq::core::PrefetchPolicy::configure (const decaf::util::Properties & *properties*) [virtual]

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix cms.PrefetchPolicy.XXX where XXX is the name of a property with a public setter method. For instance cms.PrefetchPolicy.topicPrefetch will be used to set the value of the topic prefetch limit.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

Parameters

properties	The Properties object used to configure this object.

Exceptions

NumberFormatException	if a property that is numeric cannot be converted
IllegalArgumentException	if a property can't be converted to the correct type.

6.401.3.3 virtual int activemq::core::PrefetchPolicy::getDurableTopicPrefetch ( ) const [pure virtual]

Gets the amount of messages to prefetch for a Durable Topic.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 888).

6.401.3.4 virtual int activemq::core::PrefetchPolicy::getMaxPrefetchLimit (int *value*) const [pure virtual]

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

Returns

the allowable value for a prefetch limit, either requested or the max.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 888).

6.401.3.5 virtual int activemq::core::PrefetchPolicy::getQueueBrowserPrefetch ( ) const [pure virtual]

Gets the amount of messages to prefetch for a Queue Browser.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 889).

6.401.3.6 virtual int activemq::core::PrefetchPolicy::getQueuePrefetch ( ) const [pure virtual]

Gets the amount of messages to prefetch for a Queue.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 889).

6.401.3.7 virtual int activemq::core::PrefetchPolicy::getTopicPrefetch ( ) const [pure virtual]

Gets the amount of messages to prefetch for a Topic.

Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 889).

6.401.3.8 virtual void activemq::core::PrefetchPolicy::setDurableTopicPrefetch (int *value*) [pure virtual]

Sets the amount of prefetched messages for a Durable Topic.

Parameters

value	The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 889).

6.401.3.9 virtual void activemq::core::PrefetchPolicy::setQueueBrowserPrefetch (int *value*) [pure virtual]

Sets the amount of prefetched messages for a Queue Browser.

Parameters

value	The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 889).

6.401.3.10 virtual void activemq::core::PrefetchPolicy::setQueuePrefetch (int *value*) [pure virtual]

Sets the amount of prefetched messages for a Queue.

Parameters

value	The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 890).

6.401.3.11 virtual void activemq::core::PrefetchPolicy::setTopicPrefetch (int *value*) [pure virtual]

Sets the amount of prefetched messages for a Topic.

Parameters

value	The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 890).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**PrefetchPolicy.h**

6.402 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** (int index) **const**

  Gets the Boolean value at the specified index.
- virtual void **setBool** (int index, bool value)

  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** (int index) **const**

  Gets the Byte value at the specified index.

- virtual void **setByte** (int index, unsigned char value)

  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** (int index) **const**

  Gets the Character value at the specified index.
- virtual void **setChar** (int index, char value)

  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** (int index) **const**

  Gets the Short value at the specified index.
- virtual void **setShort** (int index, short value)

  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** (int index) **const**

  Gets the Integer value at the specified index.
- virtual void **setInt** (int index, int value)

  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** (int index) **const**

  Gets the Long value at the specified index.
- virtual void **setLong** (int index, long long value)

  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** (int index) **const**

  Gets the Float value at the specified index.
- virtual void **setFloat** (int index, float value)

  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** (int index) **const**

  Gets the Double value at the specified index.
- virtual void **setDouble** (int index, double value)

  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** (int index) **const**

  Gets the String value at the specified index.
- virtual void **setString** (int index, **const** std::string &value)

  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** (int index) **const**

  Gets the Byte Array value at the specified index.
- virtual void **setByteArray** (int index, **const** std::vector< unsigned char > &value)

  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.

6.402.1 Detailed Description

List of primitives.

6.402.2 Constructor & Destructor Documentation

6.402.2.1 activemq::util::PrimitiveList::PrimitiveList ( )

Default Constructor, creates an Empty list.

6.402.2.2 virtual activemq::util::PrimitiveList::~PrimitiveList ( ) [virtual]

6.402.2.3 activemq::util::PrimitiveList::PrimitiveList (const decaf::util::List< PrimitiveValueNode > & *src*)

Copy Constructor.

Parameters

src	- the Decaf List of PrimtiveNodeValues to copy

6.402.2.4 activemq::util::PrimitiveList::PrimitiveList (const PrimitiveList & *src*)

Copy Constructor.

Parameters

src	- the **PrimitiveList** (p. 1638) to copy

6.402.3 Member Function Documentation

6.402.3.1 virtual bool activemq::util::PrimitiveList::getBool (int *index*) const [virtual]

Gets the Boolean value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.2 virtual unsigned char activemq::util::PrimitiveList::getByte (int *index*) const [virtual]

Gets the Byte value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.3 virtual std::vector<unsigned char> activemq::util::PrimitiveList::getByteArray (int *index*) const [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

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.4 virtual char activemq::util::PrimitiveList::getChar (int *index*) const [virtual]

Gets the Character value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.5 virtual double activemq::util::PrimitiveList::getDouble (int *index*) const [virtual]

Gets the Double value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.6 virtual float activemq::util::PrimitiveList::getFloat (int *index*) const [virtual]

Gets the Float value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.7 virtual int activemq::util::PrimitiveList::getInt (int *index*) const [virtual]

Gets the Integer value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.8 virtual long long activemq::util::PrimitiveList::getLong (int *index*) const [virtual]

Gets the Long value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.9 virtual short activemq::util::PrimitiveList::getShort (int *index*) const [virtual]

Gets the Short value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.10 virtual std::string activemq::util::PrimitiveList::getString (int *index*) const [virtual]

Gets the String value at the specified index.

Parameters

index	- index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBounds-Exception	if index is > **size()** (p. 1286)
UnsupportedOperation-Exception	if the type at index is not of the type that this method is to return or can convert to.

6.402.3.11 virtual void activemq::util::PrimitiveList::setBool (int *index*, bool *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.12 virtual void activemq::util::PrimitiveList::setByte (int *index*, unsigned char *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.13 virtual void activemq::util::PrimitiveList::setByteArray (int *index*, const std::vector< unsigned char > & *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.14 virtual void activemq::util::PrimitiveList::setChar (int *index*, char *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.15 virtual void activemq::util::PrimitiveList::setDouble (int *index*, double *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.16 virtual void activemq::util::PrimitiveList::setFloat (int *index*, float *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.17 virtual void activemq::util::PrimitiveList::setInt (int *index*, int *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.18 virtual void activemq::util::PrimitiveList::setLong (int *index*, long long *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.19 virtual void activemq::util::PrimitiveList::setShort (int *index*, short *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.20 virtual void activemq::util::PrimitiveList::setString (int *index*, const std::string & *value*) [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

IndexOutOfBounds-Exception	if index > **size()** (p. 1286).

6.402.3.21 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**

6.403 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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  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.

6.403.1 Detailed Description

Map of named primitives.

6.403.2 Constructor & Destructor Documentation

6.403.2.1 activemq::util::PrimitiveMap::PrimitiveMap ( )

Default Constructor, creates an empty map.

6.403.2.2 virtual activemq::util::PrimitiveMap::~PrimitiveMap ( ) [virtual]

6.403.2.3 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.

6.403.2.4 activemq::util::PrimitiveMap::PrimitiveMap (const PrimitiveMap & *source*)

Copy Constructor.

Parameters

source	The **PrimitiveMap** (p. 1647) whose elements will be copied into this Map.

6.403.3 Member Function Documentation

6.403.3.1 virtual bool activemq::util::PrimitiveMap::getBool (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.2 virtual unsigned char activemq::util::PrimitiveMap::getByte (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.3 virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getByteArray (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.4 virtual char activemq::util::PrimitiveMap::getChar (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.5 virtual double activemq::util::PrimitiveMap::getDouble (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.6 virtual float activemq::util::PrimitiveMap::getFloat (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.7 virtual int activemq::util::PrimitiveMap::getInt (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.8 virtual long long activemq::util::PrimitiveMap::getLong (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.9 virtual short activemq::util::PrimitiveMap::getShort (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.10 virtual std::string activemq::util::PrimitiveMap::getString (const std::string & *key*) const [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.
UnSupportedOperation-Exception	if the value cannot be converted to the type this method returns

6.403.3.11 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.

6.403.3.12 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.

6.403.3.13 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.

6.403.3.14 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.

6.403.3.15 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.

6.403.3.16 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.

6.403.3.17 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.

6.403.3.18 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.

6.403.3.19 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.

6.403.3.20 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.

6.403.3.21 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**

6.404 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 > &buffer)

  Marshal a primitive map object to the given byte buffer.
- static void **unmarshal** (**util::PrimitiveMap** ∗map, **const** std::vector< unsigned char > &buffer)

  Unmarshal a PrimitiveMap from the provided Byte buffer.
- static void **marshal** (**const util::PrimitiveList** ∗list, std::vector< unsigned char > &buffer)

  Marshal a primitive list object to the given byte buffer.
- static void **unmarshal** (**util::PrimitiveList** ∗list, **const** std::vector< unsigned char > &buffer)

  Unmarshal a PrimitiveList from the provided byte buffer.
- static void **marshalMap** (**const util::PrimitiveMap** ∗map, **decaf::io::DataOutputStream** &dataOut)

  Marshal a primitive map object to the given DataOutputStream.
- static **util::PrimitiveMap** ∗ **unmarshalMap** (**decaf::io::DataInputStream** &dataIn)

  Unmarshal a PrimitiveMap from the provided DataInputStream.
- static void **marshalList** (**const util::PrimitiveList** ∗list, **decaf::io::DataOutputStream** &dataOut)

  Marshal a PrimitiveList to the given DataOutputStream.
- static **util::PrimitiveList** ∗ **unmarshalList** (**decaf::io::DataInputStream** &dataIn)

  Unmarshal a PrimitiveList from the given DataInputStream.

Static Protected Member Functions

- static void **marshalPrimitiveMap** (**decaf::io::DataOutputStream** &dataOut, **const decaf::util::Map**< std::string, **util::PrimitiveValueNode** > &map)

  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)

  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)

  Used to Marshal the Primitive types out on the Wire.
- static void **unmarshalPrimitiveMap** (**decaf::io::DataInputStream** &dataIn, **util::PrimitiveMap** &map)

  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::LinkedList**< **util::PrimitiveValueNode** > &list)

  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)

  Unmarshals a Primitive Type from the stream, and returns it as a value Node.

6.404.1 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.

6.404.2 Constructor & Destructor Documentation

6.404.2.1 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller ( ) [inline]

6.404.2.2 virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~PrimitiveTypesMarshaller ( ) [inline, virtual]

6.404.3 Member Function Documentation

6.404.3.1 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveMap * *map,* std::vector< unsigned char > & *buffer*) [static]

Marshal a primitive map object to the given byte buffer.

Parameters

map	Map to Marshal.
buffer	The byte buffer to write the marshaled data to.

Exceptions

Exception	if an error occurs during the marshaling process.

6.404.3.2 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveList * *list,* std::vector< unsigned char > & *buffer*) [static]

Marshal a primitive list object to the given byte buffer.

Parameters

map	The PrimitiveList to Marshal.
buffer	The byte buffer to write the marshaled data to.

Exceptions

Exception	if an error occurs during the marshaling process.

6.404.3.3 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalList (const util::PrimitiveList * *list,* decaf::io::DataOutputStream & *dataOut*) [static]

Marshal a PrimitiveList to the given DataOutputStream.

Parameters

list	The list object to Marshal
dataOut	Reference to a DataOutputStream to write the marshaled data to.

Exceptions

Exception	if an error occurs during the marshaling process.

6.404.3.4 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalMap (const util::PrimitiveMap * *map*, decaf::io::DataOutputStream & *dataOut*) [static]

Marshal a primitive map object to the given DataOutputStream.

Parameters

map	Map to Marshal.
dataOut	Reference to a DataOutputStream to write the marshaled data to.

Exceptions

Exception	if an error occurs during the marshaling process.

6.404.3.5 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive (decaf::io::DataOutputStream & *dataOut*, const util::PrimitiveValueNode & *value*) [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	if an I/O error occurs during this operation.

6.404.3.6 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive-List (decaf::io::DataOutputStream & *dataOut*, const decaf::util::List< util::PrimitiveValueNode > & *list*) [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	if an I/O error occurs during this operation.

6.404.3.7 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal-PrimitiveMap (decaf::io::DataOutputStream & *dataOut*, const decaf::util::Map< std::string, util::PrimitiveValueNode > & *map*) [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	if an I/O error occurs during this operation.

6.404.3.8 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveMap * *map*, const std::vector< unsigned char > & *buffer*) [static]

Unmarshal a PrimitiveMap from the provided Byte buffer.

Parameters

map	The Map to populate with values from the marshaled data.
buffer	The byte buffer containing the marshaled Map.

Exceptions

Exception	if an error occurs during the unmarshal process.

6.404.3.9 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveList * *list*, const std::vector< unsigned char > & *buffer*) [static]

Unmarshal a PrimitiveList from the provided byte buffer.

Parameters

map	The List to populate with values from the marshaled data.
buffer	The byte buffer containing the marshaled Map.

Exceptions

Exception	if an error occurs during the unmarshal process.

6.404.3.10 static util::PrimitiveList* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalList (decaf::io::DataInputStream & *dataIn*) [static]

Unmarshal a PrimitiveList from the given DataInputStream.

Parameters

dataIn	The DataInputStream instance to read the marshaled PrimitiveList from.

Returns

a pointer to a newly allocated PrimitiveList instnace.

Exceptions

Exception	if an error occurs during the unmarshal process.

6.404.3.11 static util::PrimitiveMap* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalMap (decaf::io::DataInputStream & *dataIn*) [static]

Unmarshal a PrimitiveMap from the provided DataInputStream.

Parameters

dataIn	The DataInputStream instance to read the marshaled PrimitiveMap from.

Returns

a pointer to a newly allocated PrimitiveMap instnace.

Exceptions

Exception	if an error occurs during the unmarshal process.

6.404.3.12 static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive (decaf::io::DataInputStream & *dataIn*) [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	if an I/O error occurs during this operation.

6.404.3.13 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveList (decaf::io::DataInputStream & *dataIn,* decaf::util::LinkedList< util::PrimitiveValueNode > & *list*) [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	if an I/O error occurs during this operation.

6.404.3.14 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveMap (decaf::io::DataInputStream & *dataIn,* util::PrimitiveMap & *map*) [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	if an I/O error occurs during this operation.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**PrimitiveTypesMarshaller.h**

6.405 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**

6.405.1 Detailed Description

Define a union type comprised of the various types.

6.405.2 Field Documentation

6.405.2.1 bool activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue

6.405.2.2 std::vector<unsigned char>* activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue

6.405.2.3 unsigned char activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue

6.405.2.4 char activemq::util::PrimitiveValueNode::PrimitiveValue::charValue

6.405.2.5 double activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue

6.405.2.6 float activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue

6.405.2.7 int activemq::util::PrimitiveValueNode::PrimitiveValue::intValue

6.405.2.8 decaf::util::List<PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::listValue

6.405.2.9 long long activemq::util::PrimitiveValueNode::PrimitiveValue::longValue

6.405.2.10 decaf::util::Map<std::string, PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue

6.405.2.11 short activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue

6.405.2.12 std::string* activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue

The documentation for this union was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

6.406 activemq::util::PrimitiveValueConverter Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 1662) 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**

6.406.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 1662) 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

6.406.2 Constructor & Destructor Documentation

6.406.2.1 activemq::util::PrimitiveValueConverter::PrimitiveValueConverter ( ) [inline]

6.406.2.2 virtual activemq::util::PrimitiveValueConverter::~PrimitiveValueConverter ( ) [inline, virtual]

6.406.3 Member Function Documentation

6.406.3.1 std::vector< unsigned char > activemq::util::PrimitiveValueConverter::convert (const PrimitiveValueNode & *value*) const [inline]

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueConverter.h**

6.407 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)

  Primtive List Constructor.
- **PrimitiveValueNode** (**const decaf::util::Map**< std::string, **PrimitiveValueNode** > &value)

  Primtive 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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  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**

  Gets the Primitive Map value of this Node.
- std::string **toString** () **const**

  Creates a string representation of this value.

6.407.1 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.

6.407.2 Member Enumeration Documentation

6.407.2.1 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

6.407.3 Constructor & Destructor Documentation

6.407.3.1 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( )

Default Constructor, creates a value of the NULL_TYPE.

6.407.3.2 activemq::util::PrimitiveValueNode::PrimitiveValueNode (bool *value*)

Boolean Value Constructor.

Parameters

value	- the new value to store.

6.407.3.3 activemq::util::PrimitiveValueNode::PrimitiveValueNode (unsigned char *value*)

Byte Value Constructor.

Parameters

value	- the new value to store.

6.407.3.4 activemq::util::PrimitiveValueNode::PrimitiveValueNode (char *value*)

Char Value Constructor.

Parameters

value	- the new value to store.

6.407.3.5 activemq::util::PrimitiveValueNode::PrimitiveValueNode (short *value*)

Short Value Constructor.

Parameters

value	- the new value to store.

6.407.3.6 activemq::util::PrimitiveValueNode::PrimitiveValueNode (int *value*)

Int Value Constructor.

Parameters

value	- the new value to store.

6.407.3.7 activemq::util::PrimitiveValueNode::PrimitiveValueNode (long long *value*)

Long Value Constructor.

Parameters

value	- the new value to store.

6.407.3.8 activemq::util::PrimitiveValueNode::PrimitiveValueNode (float *value*)

Float Value Constructor.

Parameters

value	- the new value to store.

6.407.3.9 activemq::util::PrimitiveValueNode::PrimitiveValueNode (double *value*)

Double Value Constructor.

Parameters

value	- the new value to store.

6.407.3.10 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const char * *value*)

String Value Constructor.

Parameters

value	- the new value to store.

6.407.3.11 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::string & *value*)

String Value Constructor.

Parameters

value	- the new value to store.

6.407.3.12 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::vector< unsigned char > & *value*)

Byte Array Value Constructor.

Parameters

value	- the new value to store.

6.407.3.13 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::List< PrimitiveValueNode > & *value*)

Primtive List Constructor.

Parameters

value	- the new value to store.

6.407.3.14 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::Map< std::string, PrimitiveValueNode > & *value*)

Primtive Map Value Constructor.

Parameters

value	- the new value to store.

6.407.3.15 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const PrimitiveValueNode & *node*)

Copy constructor.

Parameters

node	The instance of another node to copy to this one.

6.407.3.16 activemq::util::PrimitiveValueNode::~PrimitiveValueNode ( ) [inline]

6.407.4 Member Function Documentation

6.407.4.1 void activemq::util::PrimitiveValueNode::clear ( )

Clears the value from this wrapper converting it back to a blank NULL_TYPE value.

6.407.4.2 bool activemq::util::PrimitiveValueNode::getBool ( ) const

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.

6.407.4.3 unsigned char activemq::util::PrimitiveValueNode::getByte ( ) const

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.

6.407.4.4 std::vector<unsigned char> activemq::util::PrimitiveValueNode::getByteArray ( ) const

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.

6.407.4.5 char activemq::util::PrimitiveValueNode::getChar ( ) const

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.

6.407.4.6 double activemq::util::PrimitiveValueNode::getDouble ( ) const

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.

6.407.4.7 float activemq::util::PrimitiveValueNode::getFloat ( ) const

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.

6.407.4.8 int activemq::util::PrimitiveValueNode::getInt ( ) const

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.

6.407.4.9 const decaf::util::List<PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getList ( ) const

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.

6.407.4.10 long long activemq::util::PrimitiveValueNode::getLong ( ) const

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.

6.407.4.11 const decaf::util::Map<std::string, PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getMap ( ) const

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.

6.407.4.12 short activemq::util::PrimitiveValueNode::getShort ( ) const

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.

6.407.4.13 std::string activemq::util::PrimitiveValueNode::getString ( ) const

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.

6.407.4.14 PrimitiveType activemq::util::PrimitiveValueNode::getType ( ) const [inline]

Gets the Value Type of this type wrapper.

Returns

the PrimitiveType value for this wrapper.

6.407.4.15 PrimitiveValue activemq::util::PrimitiveValueNode::getValue ( ) const [inline]

Gets the internal Primitive Value object from this wrapper.

Returns

a copy of the contained **PrimitiveValue** (p. 1660)

6.407.4.16 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.

6.407.4.17 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.

6.407.4.18 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

6.407.4.19 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

6.407.4.20 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

6.407.4.21 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

6.407.4.22 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

6.407.4.23 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

6.407.4.24 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

6.407.4.25 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

6.407.4.26 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

6.407.4.27 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

6.407.4.28 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

6.407.4.29 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

6.407.4.30 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.

6.407.4.31 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**

6.408 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.

6.408.1 Detailed Description

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

6.408.2 Constructor & Destructor Documentation

6.408.2.1 virtual decaf::security::Principal::~Principal ( ) [inline, virtual]

6.408.3 Member Function Documentation

6.408.3.1 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.

6.408.3.2 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. 2260).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Principal.h**

6.409 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. 1723) with the default initial capacity.*
- **PriorityQueue** (int initialCapacity)

  *Creates a Priority **Queue** (p. 1723) with the capacity value supplied.*
- **PriorityQueue** (int initialCapacity, **Comparator**< E > *__comparator__)

  *Creates a Priority **Queue** (p. 1723) with the default initial capacity.*
- **PriorityQueue** (**const Collection**< E > &source)

  *Creates a **PriorityQueue** (p. 1674) containing the elements in the specified **Collection** (p. 660).*
- **PriorityQueue** (**const PriorityQueue**< E > &source)

  *Creates a **PriorityQueue** (p. 1674) containing the elements in the specified priority queue.*
- virtual ~**PriorityQueue** ()
- **PriorityQueue**< E > & **operator=** (**const Collection**< E > &source)

  *Assignment operator, assign another **Collection** (p. 660) to this one.*
- **PriorityQueue**< E > & **operator=** (**const PriorityQueue**< E > &source)

  *Assignment operator, assign another **PriorityQueue** (p. 1674) to this one.*
- virtual **decaf::util::Iterator**
  < E > * **iterator** ()
- virtual **decaf::util::Iterator**
  < E > * **iterator** () **const**
- virtual int **size** () **const**

  Returns the number of elements in this collection.
- virtual void **clear** ()

  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. 1210) 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

UnsupportedOperation-Exception	*if the clear operation is not supported by this collection*

  This implementation repeatedly invokes poll until it returns false.
- virtual bool **offer** (**const** E &value)

  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** ()

Gets and removes the element in the head of the queue.
*Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.*

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException (p. 1537)	*if there is no element in the queue.*

This implementation returns the result of poll unless the queue is empty.

- virtual bool **remove** (**const** E &value)

Removes a single instance of the specified element from the collection.
More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this **Collection** (p. 660).*

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

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.

- virtual bool **add** (**const** E &value)

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. 660) 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	*The reference to the element to add to this **Collection** (p. 660).*

Returns

*true if the element was added to this **Collection** (p. 660).*

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

This implementation returns true if offer succeeds, else throws an IllegalStateException.

- **decaf::lang::Pointer**
  < Comparator< E > > comparator () **const**

*obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 1674) is using to compare the elements in the queue with.*

Friends

- class **PriorityQueueIterator**

6.409.1 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. 689) 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. 660) and **Iterator** (p. 1209) interfaces. The **Iterator** (p. 1209) provided in method **iterator()** (p. 1680) 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. 1674) 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 enqueing and dequeing methods (offer, poll, **remove()** (p. 1681) 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

6.409.2 Constructor & Destructor Documentation

6.409.2.1 template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( ) [inline]

Creates a Priority **Queue** (p. 1723) with the default initial capacity.

6.409.2.2 template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (int *initialCapacity*) [inline]

Creates a Priority **Queue** (p. 1723) with the capacity value supplied.

Parameters

initialCapacity	The initial number of elements allocated to this **PriorityQueue** (p. 1674).

6.409.2.3 template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (int *initialCapacity,* Comparator< E > * *comparator*) [inline]

Creates a Priority **Queue** (p. 1723) with the default initial capacity.

This new **PriorityQueue** (p. 1674) takes ownership of the passed **Comparator** (p. 689) instance and uses that to determine the ordering of the elements in the **Queue** (p. 1723).

Parameters

initialCapacity	The initial number of elements allocated to this **PriorityQueue** (p. 1674).
comparator	The **Comparator** (p. 689) instance to use in sorting the elements in the **Queue** (p. 1723).

Exceptions

NullPointerException	if the passed **Comparator** (p. 689) is NULL.

6.409.2.4 template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const Collection< E > & *source*) [inline]

Creates a **PriorityQueue** (p. 1674) containing the elements in the specified **Collection** (p. 660).

Parameters

source	the **Collection** (p. 660) whose elements are to be placed into this priority queue

6.409.2.5 template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const PriorityQueue< E > & *source*) [inline]

Creates a **PriorityQueue** (p. 1674) 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

6.409.2.6 template<typename E> virtual decaf::util::PriorityQueue< E >::~PriorityQueue ( ) [inline, virtual]

6.409.3 Member Function Documentation

6.409.3.1 template<typename E> virtual bool decaf::util::PriorityQueue< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

This implementation returns true if offer succeeds, else throws an IllegalStateException.

This implementation returns true if offer succeeds, else throws an IllegalStateException.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 109).

References DECAF_CATCH_EXCEPTION_CONVERT, DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::PriorityQueue< E >::offer().

6.409.3.2 template<typename E> virtual void decaf::util::PriorityQueue< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

This implementation repeatedly invokes poll until it returns false.

This implementation repeatedly invokes poll until it returns false.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 110).

6.409.3.3 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. 1674) 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. 689) Pointer being used by this **Queue** (p. 1723).

6.409.3.4 template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1207).

References decaf::util::PriorityQueue< E >::PriorityQueueIterator.

6.409.3.5 template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator ( ) const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1208).

6.409.3.6 template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer (const E & *value*) [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. 1723) implementation does not allow Null values to be inserted into the **Queue** (p. 1723).
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 1724).

Referenced by decaf::util::PriorityQueue< E >::add().

6.409.3.7 template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue< E >::operator= (const Collection< E > & *source*) [inline]

Assignment operator, assign another **Collection** (p. 660) to this one.

Parameters

source	The **Collection** (p. 660) to copy to this one.

6.409.3.8 template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue< E >::operator= (const PriorityQueue< E > & *source*) [inline]

Assignment operator, assign another **PriorityQueue** (p. 1674) to this one.

Parameters

source	The **PriorityQueue** (p. 1674) to copy to this one.

6.409.3.9 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. 1725).

References decaf::util::AbstractCollection< E >::isEmpty().

6.409.3.10 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. 1723) 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. 1725).

References decaf::util::AbstractCollection< E >::isEmpty().

6.409.3.11 template<typename E> virtual E decaf::util::PriorityQueue< E >::remove ( ) [inline, virtual]

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException *(p. 1537)*	if there is no element in the queue.

This implementation returns the result of poll unless the queue is empty.

This implementation returns the result of poll unless the queue is empty.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 111).

References decaf::util::AbstractCollection< E >::isEmpty().

6.409.3.12 template<typename E> virtual bool decaf::util::PriorityQueue< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 92).

6.409.3.13 template<typename E> virtual int 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. 669).

6.409.4 Friends And Related Function Documentation

6.409.4.1 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/**PriorityQueue.h**

6.410 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 **DataStructure*** (p. *877) Type as defined in CommandTypes.h.*
- virtual **ProducerAck** * **cloneDataStructure** () **const**

  Clone this obbject 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. *877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure*** (p. *877) 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)

  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 Attributes

- **Pointer**< **ProducerId** > **producerId**
- int **size**

6.410.1 Constructor & Destructor Documentation

6.410.1.1 activemq::commands::ProducerAck::ProducerAck ( )

6.410.1.2 virtual activemq::commands::ProducerAck::~ProducerAck ( ) [virtual]

6.410.2 Member Function Documentation

6.410.2.1 virtual ProducerAck* activemq::commands::ProducerAck::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.410.2.2 virtual void activemq::commands::ProducerAck::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. 382).

6.410.2.3 virtual bool activemq::commands::ProducerAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.410.2.4 virtual unsigned char activemq::commands::ProducerAck::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.410.2.5 virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId ( ) const [virtual]

6.410.2.6 virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId ( ) [virtual]

6.410.2.7 virtual int activemq::commands::ProducerAck::getSize ( ) const [virtual]

6.410.2.8 virtual bool activemq::commands::ProducerAck::isProducerAck ( ) const [inline, virtual]

Returns

an answer of true to the **isProducerAck()** (p. 1685) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.410.2.9 virtual void activemq::commands::ProducerAck::setProducerId (const Pointer< ProducerId > & *producerId*) [virtual]

6.410.2.10 virtual void activemq::commands::ProducerAck::setSize (int *size*) [virtual]

6.410.2.11 virtual std::string activemq::commands::ProducerAck::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.410.2.12 virtual Pointer<Command> activemq::commands::ProducerAck::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.410.3 Field Documentation

6.410.3.1 const unsigned char activemq::commands::ProducerAck::ID_PRODUCERACK = 19 [static]

6.410.3.2 Pointer<ProducerId> activemq::commands::ProducerAck::producerId [protected]

6.410.3.3 int activemq::commands::ProducerAck::size [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerAck.h**

6.411 activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 1686).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ProducerAckMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual ∼**ProducerAckMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.411.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 1686).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.411.2 Constructor & Destructor Documentation

6.411.2.1 activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]

6.411.2.2 virtual activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::∼ProducerAckMarshaller ( ) [inline, virtual]

6.411.3 Member Function Documentation

6.411.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.411.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.411.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.411.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.411.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.411.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.411.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ProducerAckMarshaller.h**

6.412 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** () throw ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer)=0

  Execute an action given a session and producer.

6.412.1 Detailed Description

Callback for sending a message to a CMS destination.

6.412.2 Constructor & Destructor Documentation

6.412.2.1 virtual activemq::cmsutil::ProducerCallback::~ProducerCallback ( ) throw () [inline, virtual]

6.412.3 Member Function Documentation

6.412.3.1 virtual void activemq::cmsutil::ProducerCallback::doInCms (cms::Session * *session*, cms::MessageProducer * *producer*) [pure virtual]

Execute an action given a session and producer.

Parameters

session	the CMS `Session`
producer	the CMS `Producer`

Exceptions

cms::CMSException *(p. 640)*	if thrown by CMS API methods

Implemented in **activemq::cmsutil::CmsTemplate::SendExecutor** (p. 1815).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ProducerCallback.h**

6.413 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** () throw ()
- virtual void **doInCms** (**cms::Session** ∗session)

  Execute any number of operations against the supplied CMS session.
- virtual **cms::Destination** ∗ **getDestination** (**cms::Session** ∗session AMQCPP_UNUSED)

Protected Attributes

- **ProducerCallback** ∗ **action**
- **CmsTemplate** ∗ **parent**
- **cms::Destination** ∗ **destination**

6.413.1 Constructor & Destructor Documentation

6.413.1.1 activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor (ProducerCallback ∗ *action,* CmsTemplate ∗ *parent,* cms::Destination ∗ *destination*) [inline]

6.413.1.2 virtual activemq::cmsutil::CmsTemplate::ProducerExecutor::∼ProducerExecutor ( ) throw () [inline, virtual]

6.413.2 Member Function Documentation

6.413.2.1 virtual void activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms (cms::Session ∗ *session*) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session	the CMS `Session`

Exceptions

CMSException	if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 1842).

6.413.2.2 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) [inline, virtual]

6.413.3 Field Documentation

6.413.3.1 ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action [protected]

6.413.3.2 cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination [protected]

6.413.3.3 CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.414 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)
- **ProducerId** (std::string producerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ProducerId** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*

- **const Pointer< SessionId > & getParentId** () **const**
- void **setProducerSessionKey** (std::string sessionKey)
- 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 long long **getSessionId** () **const**
- virtual void **setSessionId** (long long **sessionId**)
- virtual int **compareTo** (**const ProducerId** &**value**) **const**
- virtual bool **equals** (**const ProducerId** &**value**) **const**
- virtual bool **operator==** (**const ProducerId** &**value**) **const**
- virtual bool **operator<** (**const ProducerId** &**value**) **const**
- **ProducerId** & **operator=** (**const ProducerId** &other)

Static Public Attributes

- static **const** unsigned char **ID_PRODUCERID** = 123

Protected Attributes

- std::string **connectionId**
- long long **value**
- long long **sessionId**

6.414.1 Member Typedef Documentation

6.414.1.1 typedef decaf::lang::PointerComparator<ProducerId> activemq::commands::ProducerId::COMPARATOR

6.414.2 Constructor & Destructor Documentation

6.414.2.1 activemq::commands::ProducerId::ProducerId ( )

6.414.2.2 activemq::commands::ProducerId::ProducerId (const ProducerId & *other*)

6.414.2.3 activemq::commands::ProducerId::ProducerId (const SessionId & *sessionId,* long long *consumerId*)

6.414.2.4 activemq::commands::ProducerId::ProducerId (std::string *producerId*)

6.414.2.5 virtual activemq::commands::ProducerId::~ProducerId ( ) [virtual]

6.414.3 Member Function Documentation

6.414.3.1 virtual ProducerId* activemq::commands::ProducerId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.414.3.2 virtual int activemq::commands::ProducerId::compareTo (const ProducerId & *value*) const [virtual]

6.414.3.3 virtual void activemq::commands::ProducerId::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. 879).

6.414.3.4 virtual bool activemq::commands::ProducerId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.414.3.5 virtual bool activemq::commands::ProducerId::equals (const ProducerId & *value*) const [virtual]

6.414.3.6 virtual const std::string& activemq::commands::ProducerId::getConnectionId ( ) const [virtual]

6.414.3.7 virtual std::string& activemq::commands::ProducerId::getConnectionId ( ) [virtual]

6.414.3.8 virtual unsigned char activemq::commands::ProducerId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.414.3.9 const Pointer<SessionId>& activemq::commands::ProducerId::getParentId ( ) const

6.414.3.10 virtual long long activemq::commands::ProducerId::getSessionId ( ) const [virtual]

6.414.3.11 virtual long long activemq::commands::ProducerId::getValue ( ) const [virtual]

6.414.3.12 virtual bool activemq::commands::ProducerId::operator< (const ProducerId & *value*) const [virtual]

6.414.3.13 ProducerId& activemq::commands::ProducerId::operator= (const ProducerId & *other*)

6.414.3.14 virtual bool activemq::commands::ProducerId::operator== (const ProducerId & *value*) const [virtual]

6.414.3.15 virtual void activemq::commands::ProducerId::setConnectionId (const std::string & *connectionId*) [virtual]

6.414.3.16 void activemq::commands::ProducerId::setProducerSessionKey (std::string *sessionKey*)

6.414.3.17 virtual void activemq::commands::ProducerId::setSessionId (long long *sessionId*) [virtual]

6.414.3.18 virtual void activemq::commands::ProducerId::setValue (long long *value*) [virtual]

6.414.3.19 virtual std::string activemq::commands::ProducerId::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.414.4 Field Documentation

6.414.4.1 std::string activemq::commands::ProducerId::connectionId [protected]

6.414.4.2 const unsigned char activemq::commands::ProducerId::ID_PRODUCERID = 123 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.414.4.3 long long activemq::commands::ProducerId::sessionId [protected]

6.414.4.4 long long activemq::commands::ProducerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerId.h**

6.415 activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 1694).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ProducerIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual ~**ProducerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

Gets the DataStructureType that this class marshals/unmarshals.

- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.415.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 1694).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.415.2 Constructor & Destructor Documentation

6.415.2.1 activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::ProducerIdMarshaller ( ) [inline]

6.415.2.2 virtual activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::∼ProducerIdMarshaller ( ) [inline, virtual]

6.415.3 Member Function Documentation

6.415.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.415.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.415.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.415.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.415.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.415.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.415.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ProducerIdMarshaller.h**

6.416 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*

- virtual **ProducerInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*

- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*

- **Pointer**< **RemoveInfo** > **createRemoveCommand** () **const**
- 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)

  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 Attributes

- **Pointer**< **ProducerId** > **producerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::vector
  < **decaf::lang::Pointer**
  < **BrokerId** > > **brokerPath**
- bool **dispatchAsync**
- int **windowSize**

6.416.1 Constructor & Destructor Documentation

6.416.1.1 activemq::commands::ProducerInfo::ProducerInfo ( )

6.416.1.2 virtual activemq::commands::ProducerInfo::~ProducerInfo ( ) [virtual]

6.416.2 Member Function Documentation

6.416.2.1 virtual ProducerInfo* activemq::commands::ProducerInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.416.2.2 virtual void activemq::commands::ProducerInfo::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. 382).

6.416.2.3 Pointer<RemoveInfo> activemq::commands::ProducerInfo::createRemoveCommand ( ) const

6.416.2.4 virtual bool activemq::commands::ProducerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.416.2.5 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath ( ) const [virtual]

6.416.2.6 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath ( ) [virtual]

6.416.2.7 virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.416.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination ( ) const [virtual]

6.416.2.9 virtual Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination ( ) [virtual]

6.416.2.10 virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId ( ) const [virtual]

6.416.2.11 virtual Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId ( ) [virtual]

6.416.2.12 virtual int activemq::commands::ProducerInfo::getWindowSize ( ) const [virtual]

6.416.2.13 virtual bool activemq::commands::ProducerInfo::isDispatchAsync ( ) const [virtual]

6.416.2.14 virtual bool activemq::commands::ProducerInfo::isProducerInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isProducerInfo()** (p. 1700) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.416.2.15 virtual void activemq::commands::ProducerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.416.2.16 virtual void activemq::commands::ProducerInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.416.2.17 virtual void activemq::commands::ProducerInfo::setDispatchAsync (bool *dispatchAsync*) [virtual]

6.416.2.18 virtual void activemq::commands::ProducerInfo::setProducerId (const Pointer< ProducerId > & *producerId*) [virtual]

6.416.2.19 virtual void activemq::commands::ProducerInfo::setWindowSize (int *windowSize*) [virtual]

6.416.2.20 virtual std::string activemq::commands::ProducerInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.416.2.21 virtual Pointer<Command> activemq::commands::ProducerInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.416.3 Field Documentation

6.416.3.1 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ProducerInfo::brokerPath [protected]

6.416.3.2 Pointer<ActiveMQDestination> activemq::commands::ProducerInfo::destination [protected]

6.416.3.3 bool activemq::commands::ProducerInfo::dispatchAsync [protected]

6.416.3.4 const unsigned char activemq::commands::ProducerInfo::ID_PRODUCERINFO = 6 [static]

6.416.3.5 Pointer<ProducerId> activemq::commands::ProducerInfo::producerId [protected]

6.416.3.6 int activemq::commands::ProducerInfo::windowSize [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerInfo.h**

6.417 activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 1701).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ProducerInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual ~**ProducerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.

- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.417.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 1701).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.417.2 Constructor & Destructor Documentation

6.417.2.1 activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]

6.417.2.2 virtual activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::∼ProducerInfoMarshaller ( ) [inline, virtual]

6.417.3 Member Function Documentation

6.417.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.417.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.417.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.417.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.417.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.417.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.417.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ProducerInfoMarshaller.h**

6.418 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**

- void **setTransactionState** (**const Pointer**< **TransactionState** > &transactionState)
- **Pointer**< **TransactionState** > **getTransactionState** () **const**

6.418.1 Constructor & Destructor Documentation

6.418.1.1 activemq::state::ProducerState::ProducerState (const Pointer< ProducerInfo > & *info*)

6.418.1.2 virtual activemq::state::ProducerState::~ProducerState ( ) [virtual]

6.418.2 Member Function Documentation

6.418.2.1 const Pointer<ProducerInfo>& activemq::state::ProducerState::getInfo ( ) const [inline]

6.418.2.2 Pointer<TransactionState> activemq::state::ProducerState::getTransactionState ( ) const

6.418.2.3 void activemq::state::ProducerState::setTransactionState (const Pointer< TransactionState > & *transactionState*)

6.418.2.4 std::string activemq::state::ProducerState::toString ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ProducerState.h**

6.419 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.
- int **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.
- std::string **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.
- std::string **remove** (**const** std::string &name)

  Removes the property with the given name.
- std::vector< std::string > **propertyNames** () **const**

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

- 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. 1705) instance in NULL then this **List** (p. 1286) 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. 1705) objects are equivalent.*

- std::string **toString** () **const**

  *Formats the contents of the **Properties** (p. 1705) Object into a string that can be logged, etc.*

- void **load** (**decaf::io::InputStream** *stream)

  Reads a property list (key and element pairs) from the input byte stream.

- void **load** (**decaf::io::Reader** *reader)

  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)

  *Writes this property list (key and element pairs) in this **Properties** (p. 1705) table to the output stream in a format suitable for loading into a **Properties** (p. 1705) table using the load(InputStream) method.*

- void **store** (**decaf::io::Writer** *writer, **const** std::string &comments)

  *Writes this property list (key and element pairs) in this **Properties** (p. 1705) table to the output character stream in a format that can be read by the load(Reader) method.*

Protected Attributes

- **decaf::lang::Pointer< 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.

6.419.1 Detailed Description

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

The **Properties** (p. 1705) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 1705) instance can contain an internal **Properties** (p. 1705) list that contains default values for keys not found in the **Properties** (p. 1705) **List** (p. 1286).

The **Properties** (p. 1705) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

Since

1.0

6.419.2 Constructor & Destructor Documentation

6.419.2.1 decaf::util::Properties::Properties ( )

6.419.2.2 decaf::util::Properties::Properties (const Properties & *src*)

6.419.2.3 virtual decaf::util::Properties::~Properties ( ) [virtual]

6.419.3 Member Function Documentation

6.419.3.1 void decaf::util::Properties::clear ( )

Clears all properties from the map.

6.419.3.2 Properties* decaf::util::Properties::clone ( ) const

Clones this object.

Returns

a replica of this object.

6.419.3.3 void decaf::util::Properties::copy (const Properties & *source*)

Copies the contents of the given properties object to this one, if the given **Properties** (p. 1705) instance in NULL then this **List** (p. 1286) is not modified.

Parameters

source	The source properties object.

6.419.3.4 bool decaf::util::Properties::equals (const Properties & *source*) const

Test whether two **Properties** (p. 1705) objects are equivalent.

Two **Properties** (p. 1705) 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. 1705) object to compare this instance to.

Returns

true if the contents of the two **Properties** (p. 1705) objects are the same.

6.419.3.5 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.

6.419.3.6 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`.

6.419.3.7 bool decaf::util::Properties::hasProperty (const std::string & *name*) const

Check to see if the Property exists in the set.

Parameters

name	The property name to check for in this properties set.

Returns

true if property exists, false otherwise.

6.419.3.8 bool decaf::util::Properties::isEmpty ( ) const

Returns true if the properties object is empty.

Returns

true if empty

6.419.3.9 void decaf::util::Properties::load (decaf::io::InputStream * *stream*)

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.

6.419.3.10 void decaf::util::Properties::load (decaf::io::Reader * *reader*)

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

Properties (p. 1705) 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 affect 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 '=' or ':', 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.

6.419.3.11 Properties& decaf::util::Properties::operator= (const Properties & *src*)

Assignment Operator.

Parameters

src	The **Properties** (p. 1705) list to copy to this **List** (p. 1286).

Returns

a reference to this **List** (p. 1286) for use in chaining.

6.419.3.12 std::vector<std::string> decaf::util::Properties::propertyNames ( ) const

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

Returns

a set of keys in this property list where the key and its corresponding value are strings, including the keys in the default property list.

6.419.3.13 std::string decaf::util::Properties::remove (const std::string & *name*)

Removes the property with the given name.

Parameters

name	The name of the property to remove.

Returns

the previous value of the property if set, or empty string.

6.419.3.14 std::string 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.

Returns

the old value of the property or empty string if not set.

6.419.3.15 int decaf::util::Properties::size ( ) const

Returns

The number of **Properties** (p. 1705) in this **Properties** (p. 1705) Object.

6.419.3.16 void decaf::util::Properties::store (decaf::io::OutputStream * *out,* const std::string & *comment*)

Writes this property list (key and element pairs) in this **Properties** (p. 1705) table to the output stream in a format suitable for loading into a **Properties** (p. 1705) table using the load(InputStream) method.

Properties (p. 1705) from the defaults table of this **Properties** (p. 1705) 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.

6.419.3.17 void decaf::util::Properties::store (decaf::io::Writer * *writer,* const std::string & *comments*)

Writes this property list (key and element pairs) in this **Properties** (p. 1705) table to the output character stream in a format that can be read by the load(Reader) method.

Properties (p. 1705) from the defaults table of this **Properties** (p. 1705) 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. 882) for the current time), and a line separator as generated by the Writer.

Then every entry in this **Properties** (p. 1705) 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.

6.419.3.18 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.

6.419.3.19 std::string decaf::util::Properties::toString ( ) const

Formats the contents of the **Properties** (p. 1705) Object into a string that can be logged, etc.

Returns

string value of this object.

6.419.4 Field Documentation

6.419.4.1 decaf::lang::Pointer<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**

6.420 decaf::util::logging::PropertiesChangeListener Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 1705).

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

Public Member Functions

- virtual ~**PropertiesChangeListener** ()
- virtual void **onPropertiesReset** ()=0

  *Indicates that the **Properties** (p. 1705) have all been reset and should be considered to be back to their default values.*
- virtual void **onPropertyChanged** (**const** std::string &name, **const** std::string &oldValue, **const** std::string &newValue)=0

  Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

6.420.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 1705).

Since

1.0

6.420.2 Constructor & Destructor Documentation

6.420.2.1 virtual decaf::util::logging::PropertiesChangeListener::~PropertiesChangeListener ( ) [inline, virtual]

6.420.3 Member Function Documentation

6.420.3.1 virtual void decaf::util::logging::PropertiesChangeListener::onPropertiesReset ( ) [pure virtual]

Indicates that the **Properties** (p. 1705) have all been reset and should be considered to be back to their default values.

6.420.3.2 virtual void decaf::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, includes the name of the property that was changed along with it old and new values.

Parameters

name	The name of the Property that changed.
oldValue	The old Value of the Property.
newValue	The new Value of the Property.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**PropertiesChangeListener.h**

6.421 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 ()

6.421.1 Constructor & Destructor Documentation

6.421.1.1 decaf::net::ProtocolException::ProtocolException ( ) throw () [inline]

Default Constructor.

6.421.1.2 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

6.421.1.3 decaf::net::ProtocolException::ProtocolException (const ProtocolException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.421.1.4 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

6.421.1.5 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.

6.421.1.6 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

6.421.1.7 virtual decaf::net::ProtocolException::~ProtocolException ( ) throw () [inline, virtual]

6.421.2 Member Function Documentation

6.421.2.1 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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ProtocolException.h**

6.422 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** ()

6.422.1 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.

6.422.2 Constructor & Destructor Documentation

6.422.2.1 virtual decaf::security::PublicKey::∼PublicKey ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/**PublicKey.h**

6.423 decaf::io::PushbackInputStream Class Reference

A **PushbackInputStream** (p. 1716) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

`#include <src/main/decaf/io/PushbackInputStream.h>`

Inheritance diagram for decaf::io::PushbackInputStream:

Public Member Functions

- **PushbackInputStream** (**InputStream** ∗stream, bool **own**=false)

  *Creates a **PushbackInputStream** (p. 1716) and saves its argument, the input stream in, for later use.*

- **PushbackInputStream** (**InputStream** ∗stream, int bufSize, bool **own**=false)

  *Creates a **PushbackInputStream** (p. 1716) and saves its argument, the input stream in, for later use.*
- virtual ∼**PushbackInputStream** ()
- void **unread** (unsigned char value)

  Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.
- void **unread** (**const** unsigned char ∗buffer, int size)

  Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.
- void **unread** (**const** unsigned char ∗buffer, int size, int offset, int length)

  Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.
- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual long long **skip** (long long num)

  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.
  *The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)	*if an I/O error occurs.*
UnsupportedOperation-Exception	*if the concrete stream class does not support skipping bytes.*

- virtual void **mark** (int readLimit)

  *Does nothing except throw an **IOException** (p. 1198).*
- virtual void **reset** ()

  *Does nothing except throw an **IOException** (p. 1198).*
- virtual bool **markSupported** () **const**

  *Does nothing except throw an **IOException** (p. 1198).*

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char ∗buffer, int size, int offset, int length)

6.423.1 Detailed Description

A **PushbackInputStream** (p. 1716) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

This is useful in situations where it is convenient for a fragment of code to read an indefinite number of data bytes that are delimited by a particular byte value; after reading the terminating byte, the code fragment can "unread" it, so that the next read operation on the input stream will reread the byte that was pushed back. For example, bytes representing the characters constituting an identifier might be terminated by a byte representing an operator character; a method whose job is to read just an identifier can read until it sees the operator and then push the operator back to be re-read.

Since

1.0

6.423.2 Constructor & Destructor Documentation

6.423.2.1 decaf::io::PushbackInputStream::PushbackInputStream (InputStream * *stream,* bool *own* = `false`)

Creates a **PushbackInputStream** (p. 1716) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

Parameters

stream	The **InputStream** (p. 1134) instance to wrap.
Boolean	value indicating if this **FilterInputStream** (p. 1032) owns the wrapped stream.

6.423.2.2 decaf::io::PushbackInputStream::PushbackInputStream (InputStream * *stream,* int *bufSize,* bool *own* = `false`)

Creates a **PushbackInputStream** (p. 1716) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

Parameters

stream	The **InputStream** (p. 1134) instance to wrap.
bufSize	The number of byte to allocate for pushback into this stream.
Boolean	value indicating if this **FilterInputStream** (p. 1032) owns the wrapped stream.

Exceptions

IllegalArgumentException	if the bufSize argument is $<$ zero.

6.423.2.3 virtual decaf::io::PushbackInputStream::~PushbackInputStream ( ) [virtual]

6.423.3 Member Function Documentation

6.423.3.1 virtual int decaf::io::PushbackInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Returns the sum of the number of pushed back bytes if any and the amount of bytes available in the underlying stream via a call to available.

Reimplemented from **decaf::io::FilterInputStream** (p. 1034).

6.423.3.2 virtual int decaf::io::PushbackInputStream::doReadArrayBounded (unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.423.3.3 virtual int decaf::io::PushbackInputStream::doReadByte ( ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.423.3.4 virtual void decaf::io::PushbackInputStream::mark (int *readLimit*) [virtual]

Does nothing except throw an **IOException** (p. 1198).

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.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

Parameters

readLimit	The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.423.3.5 virtual bool decaf::io::PushbackInputStream::markSupported ( ) const [inline, virtual]

Does nothing except throw an **IOException** (p. 1198).

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.

The default implementation of this method returns false.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1035).

6.423.3.6 virtual void decaf::io::PushbackInputStream::reset ( ) [virtual]

Does nothing except throw an **IOException** (p. 1198).

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. 1198) might be thrown.
- If such an **IOException** (p. 1198) 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. 1198).
- If an **IOException** (p. 1198) 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.

The default implementation of this method throws an **IOException** (p. 1198).

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1036).

6.423.3.7 virtual long long decaf::io::PushbackInputStream::skip (long long *num*) [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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

This method first skips bytes in the local pushed back buffer before attempting to complete the request by calling the underlying stream skip method with the remainder of bytes that needs to be skipped.

Reimplemented from **decaf::io::FilterInputStream** (p. 1036).

6.423.3.8 void decaf::io::PushbackInputStream::unread (unsigned char *value*)

Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.

Parameters

value	The byte that is to be placed at the front of the push back buffer.

Exceptions

IOException *(p. 1198)*	if there is not enough space in the pushback buffer or this stream has already been closed.

6.423.3.9 void decaf::io::PushbackInputStream::unread (const unsigned char * *buffer,* int *size*)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

Parameters

buffer	The bytes to copy to the front of push back buffer.
size	The size of the array to be copied.

Exceptions

NullPointerException	if the buffer passed is NULL.
IndexOutOfBounds-Exception	if the size value given is negative.
IOException *(p. 1198)*	if there is not enough space in the pushback buffer or this stream has already been closed.

6.423.3.10 void decaf::io::PushbackInputStream::unread (const unsigned char * *buffer,* int *size,* int *offset,* int *length*)

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

Parameters

buffer	The bytes to copy to the front of push back buffer.
size	The size of the array to be copied.
offset	The position in the buffer to start copying from.
length	The number of bytes to push back from the passed buffer.

Exceptions

NullPointerException	if the buffer passed is NULL.
IndexOutOfBounds-Exception	if the offset + length is greater than the buffer size.

IOException *(p. 1198)*	if there is not enough space in the pushback buffer or this stream has already been closed.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**PushbackInputStream.h**

6.424 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** () throw ()
- virtual std::string **getQueueName** () **const** =0

  Gets the name of this queue.

6.424.1 Detailed Description

An interface encapsulating a provider-specific queue name.

Messages sent to a **Queue** (p. 1722) are sent to a Single Subscriber on that **Queue** (p. 1722) **Destination** (p. 936). 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. 1426) in a **Queue** (p. 1722) is not defined by the CMS API, consult your Provider documentation for this information.

Since

1.0

6.424.2 Constructor & Destructor Documentation

6.424.2.1 virtual cms::Queue::~Queue ( ) throw () [virtual]

6.424.3 Member Function Documentation

6.424.3.1 virtual std::string cms::Queue::getQueueName ( ) const [pure virtual]

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::commands::ActiveMQQueue** (p. 252).

The documentation for this class was generated from the following file:

- src/main/cms/**Queue.h**

6.425 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

  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

  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

  Gets but not removes the element in the head of the queue.

6.425.1 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. 1723) 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. 1723) interface the methods of this class cannot return null to indicate that a **Queue** (p. 1723) 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. 1723) 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. 1723) must be *assignable* in order to utilize these methods.

Since

1.0

6.425.2 Constructor & Destructor Documentation

6.425.2.1 template<typename E> virtual decaf::util::Queue< E >::~Queue ( ) [inline, virtual]

6.425.3 Member Function Documentation

6.425.3.1 template<typename E> virtual E decaf::util::Queue< E >::element ( ) const [pure virtual]

Gets but not removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException *(p. 1537)*	if there is no element in the queue.

Implemented in **decaf::util::LinkedList< E >** (p. 1276), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1276), **decaf::util::LinkedList< CompositeTask * >** (p. 1276), **decaf::util::LinkedList< URI >** (p. 1276), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1276), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1276), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1276), **decaf::util::LinkedList< Pointer< Command > >** (p. 1276), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1276), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1276), **decaf::util::LinkedList< cms::Destination * >** (p. 1276), **decaf::util::LinkedList< cms::Session * >** (p. 1276), **decaf::util::LinkedList< cms::Connection * >** (p. 1276), and **decaf::util::AbstractQueue< E >** (p. 111).

6.425.3.2 template<typename E> virtual bool decaf::util::Queue< E >::offer (const E & *value*) [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. 1723) implementation does not allow Null values to be inserted into the **Queue** (p. 1723).
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::LinkedBlockingQueue< E >** (p. 1262), **decaf::util::LinkedList< E >** (p. 1279), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1279), **decaf::util::LinkedList< CompositeTask * >** (p. 1279), **decaf::util::LinkedList< URI >** (p. 1279), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1279), **decaf-**

::util::LinkedList< Pointer< DestinationInfo > > (p. 1279), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1279), **decaf::util::LinkedList< Pointer< Command > >** (p. 1279), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1279), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1279), **decaf::util::LinkedList< cms::Destination * >** (p. 1279), **decaf::util::LinkedList< cms::Session * >** (p. 1279), **decaf::util::LinkedList< cms::Connection * >** (p. 1279), **decaf::util::PriorityQueue< E >** (p. 1680), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2062).

Referenced by decaf::util::AbstractQueue< E >::add().

6.425.3.3 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::concurrent::LinkedBlockingQueue< E >** (p. 1263), **decaf::util::LinkedList< E >** (p. 1280), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1280), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1280), **decaf::util::LinkedList< CompositeTask * >** (p. 1280), **decaf::util::LinkedList< URI >** (p. 1280), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1280), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1280), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1280), **decaf::util::LinkedList< Pointer< Command > >** (p. 1280), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1280), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1280), **decaf::util::LinkedList< cms::Destination * >** (p. 1280), **decaf::util::LinkedList< cms::Session * >** (p. 1280), **decaf::util::LinkedList< cms::Connection * >** (p. 1280), and **decaf::util::PriorityQueue< E >** (p. 1681).

Referenced by decaf::util::AbstractQueue< E >::element().

6.425.3.4 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. 1723) 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::concurrent::LinkedBlockingQueue< E >** (p. 1264), **decaf::util::LinkedList< E >** (p. 1281), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1281), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1281), **decaf::util::LinkedList< CompositeTask * >** (p. 1281), **decaf::util::LinkedList< URI >** (p. 1281), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1281), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1281), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1281), **decaf::util::LinkedList< Pointer< Command > >** (p. 1281), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1281), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1281), **decaf::util-**

::LinkedList< cms::Destination * > (p. 1281), **decaf::util::LinkedList< cms::Session * >** (p. 1281), **decaf::util::LinkedList< cms::Connection * >** (p. 1281), **decaf::util::PriorityQueue< E >** (p. 1681), and **decaf::util::concurrent::SynchronousQueue< E >** (p. 2063).

Referenced by decaf::util::AbstractQueue< E >::clear(), and decaf::util::AbstractQueue< E >::remove().

6.425.3.5 template<typename E> virtual E decaf::util::Queue< E >::remove ( ) [pure virtual]

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** (p. 1537) if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException *(p. 1537)*	if there is no element in the queue.

Implemented in **decaf::util::LinkedList< E >** (p. 1283), **decaf::util::LinkedList< Pointer< Transport > >** (p. 1283), **decaf::util::LinkedList< cms::MessageConsumer * >** (p. 1283), **decaf::util::LinkedList< CompositeTask * >** (p. 1283), **decaf::util::LinkedList< URI >** (p. 1283), **decaf::util::LinkedList< Pointer< MessageDispatch > >** (p. 1283), **decaf::util::LinkedList< Pointer< DestinationInfo > >** (p. 1283), **decaf::util::LinkedList< PrimitiveValueNode >** (p. 1283), **decaf::util::LinkedList< Pointer< Command > >** (p. 1283), **decaf::util::LinkedList< Pointer< BackupTransport > >** (p. 1283), **decaf::util::LinkedList< cms::MessageProducer * >** (p. 1283), **decaf::util::LinkedList< cms::Destination * >** (p. 1283), **decaf::util::LinkedList< cms::Session * >** (p. 1283), **decaf::util::LinkedList< cms::Connection * >** (p. 1283), **decaf::util::PriorityQueue< E >** (p. 1681), and **decaf::util::AbstractQueue< E >** (p. 111).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Queue.h**

6.426 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p. 1722) without removing them.

`#include <src/main/cms/QueueBrowser.h>`

Inheritance diagram for cms::QueueBrowser:

Public Member Functions

- virtual ~**QueueBrowser** () throw ()
- virtual **const Queue * getQueue** () **const** =0
- virtual std::string **getMessageSelector** () **const** =0
- virtual **cms::MessageEnumeration * getEnumeration** ()=0

  *Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 1722) in the order that a client would receive them.*

6.426.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p. 1722) without removing them.

To browse the contents of the **Queue** (p. 1722) the client calls the `getEnumeration` method to retrieve a new instance of a **Queue** (p. 1722) Enumerator. The client then calls the hasMoreMessages method of the Enumeration, if it returns true the client can safely call the nextMessage method of the Enumeration instance.

```
Enumeration* enumeration = queueBrowser->getEnumeration();

while( enumeration->hasMoreMessages() ) {
    cms::Message* message = enumeration->nextMessage();

    // ... Do something with the Message.

    delete message;
}
```

Since

1.1

6.426.2 Constructor & Destructor Documentation

6.426.2.1 virtual cms::QueueBrowser::~QueueBrowser ( ) throw () [virtual]

6.426.3 Member Function Documentation

6.426.3.1 virtual cms::MessageEnumeration* cms::QueueBrowser::getEnumeration ( ) [pure virtual]

Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 1722) in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

Returns

a pointer to a **Queue** (p. 1722) Enumeration, this Pointer is owned by the **QueueBrowser** (p. 1726) and should not be deleted by the client.

Exceptions

CMSException *(p. 640)*	if an internal error occurs.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 253).

6.426.3.2 virtual std::string cms::QueueBrowser::getMessageSelector ( ) const [pure virtual]

Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions

CMSException *(p. 640)*	if an internal error occurs.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 254).

6.426.3.3 virtual const Queue* cms::QueueBrowser::getQueue ( ) const [pure virtual]

Returns

the **Queue** (p. 1722) that this browser is listening on.

Exceptions

CMSException *(p. 640)*	if an internal error occurs.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 254).

The documentation for this class was generated from the following file:

- src/main/cms/**QueueBrowser.h**

6.427 decaf::util::Random Class Reference

Random (p. 1728) Value Generator which is used to generate a stream of pseudorandom numbers.

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

Inheritance diagram for decaf::util::Random:

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.
- virtual ∼**Random** ()
- bool **nextBoolean** ()

  Answers the next pseudo-random, uniformly distributed boolean value generated by this 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)

  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.
- virtual void **nextBytes** (std::vector< unsigned char > &buf)

  Modifies the byte array by a random sequence of bytes generated by this random number generator.
- virtual void **nextBytes** (unsigned char *buf, int size)

  Modifies the byte array by a random sequence of bytes generated by this random number generator.
- virtual 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.*

6.427.1 Detailed Description

Random (p. 1728) Value Generator which is used to generate a stream of pseudorandom numbers.

The algorithms implemented by class **Random** (p. 1728) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Since

1.0

6.427.2 Constructor & Destructor Documentation

6.427.2.1 decaf::util::Random::Random ( )

Construct a random generator with the current time of day in milliseconds as the initial state.

See also

setSeed (p. 1732)

6.427.2.2 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. 1732)

6.427.2.3 virtual decaf::util::Random::~Random ( ) [virtual]

6.427.3 Member Function Documentation

6.427.3.1 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. 1730)
nextDouble (p. 1730)
nextFloat (p. 1731)
nextInt() (p. 1731)
nextInt(int) (p. 1731)
nextGaussian (p. 1731)
nextLong (p. 1732)

Reimplemented in **decaf::security::SecureRandom** (p. 1801).

6.427.3.2 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

6.427.3.3 virtual void decaf::util::Random::nextBytes (std::vector< unsigned char > & *buf*) [virtual]

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. 1729)

Reimplemented in **decaf::security::SecureRandom** (p. 1801).

6.427.3.4 virtual void decaf::util::Random::nextBytes (unsigned char * *buf*, int *size*) [virtual]

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. 1729)

Exceptions

NullPointerException	if buff is NULL
IllegalArgumentException	if size is negative

Reimplemented in **decaf::security::SecureRandom** (p. 1802).

6.427.3.5 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. 1731)

6.427.3.6 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. 1730)

6.427.3.7 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. 1730)

6.427.3.8 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. 1729)
nextLong (p. 1732)

6.427.3.9 int decaf::util::Random::nextInt (int *n*)

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

Parameters

n	The int value that defines the max value of the return.

Returns

the next pseudo random int value.

Exceptions

IllegalArgumentException	if n is less than or equal to zero.

6.427.3.10 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. 1729)
nextInt() (p. 1731)
nextInt(int) (p. 1731)

6.427.3.11 virtual void decaf::util::Random::setSeed (unsigned long long *seed*) [virtual]

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. 1729)
Random() (p. 1729)
#Random(long)

Reimplemented in **decaf::security::SecureRandom** (p. 1802).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Random.h**

6.428 decaf::lang::Readable Class Reference

A **Readable** (p. 1732) is a source of characters.

`#include <src/main/decaf/lang/Readable.h>`

Inheritance diagram for decaf::lang::Readable:

Public Member Functions

- virtual ~**Readable** ()
- virtual int **read** (**decaf::nio::CharBuffer** *charBuffer)=0

  Attempts to read characters into the specified character buffer.

6.428.1 Detailed Description

A **Readable** (p. 1732) is a source of characters.

Characters from a **Readable** (p. 1732) are made available to callers of the read method via a CharBuffer.

Since

1.0

6.428.2 Constructor & Destructor Documentation

6.428.2.1 virtual decaf::lang::Readable::~Readable ( ) [inline, virtual]

6.428.3 Member Function Documentation

6.428.3.1 virtual int decaf::lang::Readable::read (decaf::nio::CharBuffer * *charBuffer*) [pure 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

charBuffer	The Buffer to read Characters into.

Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

Exceptions

IOException	if an I/O error occurs.
NullPointerException	if buffer is NULL.
ReadOnlyBufferException	if charBuffer is a read only buffer.

Implemented in **decaf::io::Reader** (p. 1738).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Readable.h**

6.429 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.

6.429.1 Detailed Description

Runnable class that is used by the {.

See also

InactivityMonitor (p. 1104)} class the check for timeouts related to **transport** (p. 56) reads.

Since

3.1

6.429.2 Constructor & Destructor Documentation

6.429.2.1 activemq::transport::inactivity::ReadChecker::ReadChecker (InactivityMonitor ∗ *parent*)

6.429.2.2 virtual activemq::transport::inactivity::ReadChecker::∼ReadChecker ( ) [virtual]

6.429.3 Member Function Documentation

6.429.3.1 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. 1793).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**ReadChecker.h**

6.430 decaf::io::Reader Class Reference

`#include <src/main/decaf/io/Reader.h>`

Inheritance diagram for decaf::io::Reader:

Public Member Functions

- virtual ~**Reader** ()
- virtual void **mark** (int readAheadLimit)

  Marks the present position in the stream.
- virtual bool **markSupported** () **const**

  *Tells whether this stream supports the **mark()** (p. 1736) operation.*
- virtual bool **ready** () **const**

  Tells whether this stream is ready to be read.
- virtual void **reset** ()

  Resets the stream.
- virtual long long **skip** (long long count)

  Skips characters.
- virtual int **read** (std::vector< char > &buffer)

  Reads characters into an array.
- virtual int **read** (char *buffer, int size)

  Reads characters into an array, the method will attempt to read as much data as the size of the array.
- virtual int **read** (char *buffer, int size, int offset, int length)

  Reads characters into a portion of an array.
- virtual int **read** ()

  Reads a single character.
- virtual int **read** (**decaf::nio::CharBuffer** *charBuffer)

  Attempts to read characters into the specified character buffer.

Protected Member Functions

- **Reader** ()
- virtual int **doReadArrayBounded** (char *buffer, int size, int offset, int length)=0

  Override this method to customize the functionality of the method read(unsigned char buffer, int size, int offset, int length).*
- virtual int **doReadVector** (std::vector< char > &buffer)

  *Override this method to customize the functionality of the method **read(std::vector<char>& buffer)** (p. 1736).*
- virtual int **doReadArray** (char *buffer, int length)

  Override this method to customize the functionality of the method read(char buffer, std::size_t length).*
- virtual int **doReadChar** ()

  *Override this method to customize the functionality of the method **read()** (p. 1737).*
- virtual int **doReadCharBuffer** (**decaf::nio::CharBuffer** *charBuffer)

  Override this method to customize the functionality of the method read(CharBuffer charBuffer).*

6.430.1 Constructor & Destructor Documentation

6.430.1.1 decaf::io::Reader::Reader ( ) [protected]

6.430.1.2 virtual decaf::io::Reader::~Reader ( ) [virtual]

6.430.2 Member Function Documentation

6.430.2.1 virtual int decaf::io::Reader::doReadArray (char * *buffer,* int *length*) [protected, virtual]

Override this method to customize the functionality of the method read(char* buffer, std::size_t length).

6.430.2.2 virtual int decaf::io::Reader::doReadArrayBounded (char * *buffer,* int *size,* int *offset,* int *length*) [protected, pure virtual]

Override this method to customize the functionality of the method read(unsigned char* buffer, int size, int offset, int length).

All subclasses must override this method to provide the basic **Reader** (p. 1734) functionality.

Implemented in **decaf::io::InputStreamReader** (p. 1144).

6.430.2.3 virtual int decaf::io::Reader::doReadChar ( ) [protected, virtual]

Override this method to customize the functionality of the method **read()** (p. 1737).

6.430.2.4 virtual int decaf::io::Reader::doReadCharBuffer (decaf::nio::CharBuffer * *charBuffer*) [protected, virtual]

Override this method to customize the functionality of the method read(CharBuffer* charBuffer).

6.430.2.5 virtual int decaf::io::Reader::doReadVector (std::vector< char > & *buffer*) [protected, virtual]

Override this method to customize the functionality of the method **read(std::vector<char>& buffer)** (p. 1736).

6.430.2.6 virtual void decaf::io::Reader::mark (int *readAheadLimit*) [virtual]

Marks the present position in the stream.

Subsequent calls to **reset()** (p. 1738) will attempt to reposition the stream to this point. Not all character-input streams support the **mark()** (p. 1736) operation.

Parameters

readAheadLimit	Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs, or the stream does not support mark.

6.430.2.7 virtual bool decaf::io::Reader::markSupported ( ) const [inline, virtual]

Tells whether this stream supports the **mark()** (p. 1736) operation.

The default implementation always returns false. Subclasses should override this method.

Returns

true if and only if this stream supports the mark operation.

6.430.2.8 virtual int decaf::io::Reader::read (std::vector< char > & *buffer*) [virtual]

Reads characters into an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

buffer	The buffer to read characters into.

Returns

The number of characters read, or -1 if the end of the stream has been reached

Exceptions

IOException *(p. 1198)*	thrown if an I/O error occurs.

6.430.2.9 virtual int decaf::io::Reader::read (char * *buffer,* int *size*) [virtual]

Reads characters into an array, the method will attempt to read as much data as the size of the array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

buffer	The target char buffer.
size	The size in bytes of the target buffer.

Returns

The number of bytes read or -1 if the end of stream is reached.

Exceptions

IOException *(p. 1198)*	thrown if an I/O error occurs.
NullPointerException	if buffer is NULL.

6.430.2.10 virtual int decaf::io::Reader::read (char * *buffer,* int *size,* int *offset,* int *length*) [virtual]

Reads characters into a portion of an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters

buffer	The target char buffer.
size	The size in bytes of the target buffer.
offset	The position in the buffer to start filling.
length	The maximum number of bytes to read.

Returns

The number of bytes read or -1 if the end of stream is reached.

Exceptions

IOException *(p. 1198)*	thrown if an I/O error occurs.
NullPointerException	if buffer is NULL.
IndexOutOfBounds-Exception	if the offset + length is greater than the array size.

6.430.2.11 virtual int decaf::io::Reader::read ( ) [virtual]

Reads a single character.

This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.

Subclasses that intend to support efficient single-character input should override this method.

Returns

The character read, as an integer in the range 0 to 65535 (0x00-0xffff), or -1 if the end of the stream has been reached.

Exceptions

IOException *(p. 1198)*	thrown if an I/O error occurs.

6.430.2.12 virtual int decaf::io::Reader::read (decaf::nio::CharBuffer * *charBuffer*) [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

charBuffer	The Buffer to read Characters into.

Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.
NullPointerException	if buffer is NULL.
ReadOnlyBufferException	if charBuffer is a read only buffer.

Implements **decaf::lang::Readable** (p. 1733).

6.430.2.13 virtual bool decaf::io::Reader::ready ( ) const [virtual]

Tells whether this stream is ready to be read.

Returns

True if the next **read()** (p. 1737) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

Reimplemented in **decaf::io::InputStreamReader** (p. 1144).

6.430.2.14 virtual void decaf::io::Reader::reset ( ) [virtual]

Resets the stream.

If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the **reset()** (p. 1738) operation, and some support **reset()** (p. 1738) without supporting **mark()** (p. 1736).

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

6.430.2.15 virtual long long decaf::io::Reader::skip (long long *count*) [virtual]

Skips characters.

This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

Parameters

count	The number of character to skip.

Returns

the number of Character actually skipped.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Reader.h**

6.431 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 ()

6.431.1 Constructor & Destructor Documentation

6.431.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( ) throw () [inline]

Default Constructor.

6.431.1.2 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.431.1.3 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const ReadOnlyBufferException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.431.1.4 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

6.431.1.5 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.

6.431.1.6 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

6.431.1.7 virtual decaf::nio::ReadOnlyBufferException::~ReadOnlyBufferException ( ) throw () [inline, virtual]

6.431.2 Member Function Documentation

6.431.2.1 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. 2189).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ReadOnlyBufferException.h**

6.432 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 1741) 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.

6.432.1 Detailed Description

A **ReadWriteLock** (p. 1741) 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. 1741) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 1305) 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

6.432.2 Constructor & Destructor Documentation

6.432.2.1 virtual decaf::util::concurrent::locks::ReadWriteLock::~ReadWriteLock ( ) [inline, virtual]

6.432.3 Member Function Documentation

6.432.3.1 virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock ( ) [pure virtual]

Returns the lock used for reading.

Returns

the lock used for reading.

6.432.3.2 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**

6.433 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** () throw ()
- virtual void **doInCms** (**cms::Session** ∗session)

  Execute any number of operations against the supplied CMS session.
- virtual **cms::Destination** ∗ **getDestination** (**cms::Session** ∗session AMQCPP_UNUSED)
- **cms::Message** ∗ **getMessage** ()

Protected Attributes

- **cms::Destination** ∗ **destination**
- std::string **selector**
- bool **noLocal**
- **cms::Message** ∗ **message**
- **CmsTemplate** ∗ **parent**

6.433.1 Constructor & Destructor Documentation

6.433.1.1 activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor (CmsTemplate ∗ *parent*, cms::Destination ∗ *destination*, const std::string & *selector*, bool *noLocal*) [inline]

6.433.1.2 virtual activemq::cmsutil::CmsTemplate::ReceiveExecutor::∼ReceiveExecutor ( ) throw () [inline, virtual]

6.433.2 Member Function Documentation

6.433.2.1 virtual void activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms (cms::Session * *session*) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session	the CMS `Session`

Exceptions

CMSException	if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 1842).

6.433.2.2 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) [inline, virtual]

6.433.2.3 cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage ( ) [inline]

6.433.3 Field Documentation

6.433.3.1 cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination [protected]

6.433.3.2 cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::message [protected]

6.433.3.3 bool activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal [protected]

6.433.3.4 CmsTemplate* activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent [protected]

6.433.3.5 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**

6.434 activemq::core::RedeliveryPolicy Class Reference

Interface for a **RedeliveryPolicy** (p. 1744) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

`#include <src/main/activemq/core/RedeliveryPolicy.h>`

Inheritance diagram for activemq::core::RedeliveryPolicy:

Public Member Functions

- virtual ~**RedeliveryPolicy** ()
- virtual double **getBackOffMultiplier** () **const** =0
- virtual void **setBackOffMultiplier** (double value)=0

  Sets the Back-Off Multiplier for Message Redelivery.

- virtual short **getCollisionAvoidancePercent** () **const** =0
- virtual void **setCollisionAvoidancePercent** (short value)=0
- virtual long long **getInitialRedeliveryDelay** () **const** =0

  Gets the initial time that redelivery of messages is delayed.
- virtual void **setInitialRedeliveryDelay** (long long value)=0

  Sets the initial time that redelivery will be delayed.
- virtual long long **getRedeliveryDelay** () **const** =0

  Gets the time that redelivery of messages is delayed.
- virtual void **setRedeliveryDelay** (long long value)=0

  Sets the time that redelivery will be delayed.
- virtual int **getMaximumRedeliveries** () **const** =0

  Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.
- virtual void **setMaximumRedeliveries** (int maximumRedeliveries)=0

  Sets the Maximum allowable redeliveries for a Message.
- virtual long long **getNextRedeliveryDelay** (long long previousDelay)=0

  Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.
- virtual bool **isUseCollisionAvoidance** () **const** =0
- virtual void **setUseCollisionAvoidance** (bool value)=0
- virtual bool **isUseExponentialBackOff** () **const** =0
- virtual void **setUseExponentialBackOff** (bool value)=0
- virtual **RedeliveryPolicy** ∗ **clone** () **const** =0

  Create a copy of this Policy and return it.
- virtual void **configure** (**const decaf::util::Properties** &properties)

  Checks the supplied properties object for properties matching the configurable settings of this class.

Static Public Attributes

- static **const** long long **NO_MAXIMUM_REDELIVERIES**

Protected Member Functions

- **RedeliveryPolicy** ()

6.434.1 Detailed Description

Interface for a **RedeliveryPolicy** (p. 1744) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

Since

3.2.0

6.434.2 Constructor & Destructor Documentation

6.434.2.1 activemq::core::RedeliveryPolicy::RedeliveryPolicy ( ) [protected]

6.434.2.2 virtual activemq::core::RedeliveryPolicy::∼RedeliveryPolicy ( ) [virtual]

6.434.3 Member Function Documentation

6.434.3.1 virtual RedeliveryPolicy∗ activemq::core::RedeliveryPolicy::clone ( ) const [pure virtual]

Create a copy of this Policy and return it.

Returns

pointer to a new **RedeliveryPolicy** (p. 1744) that is a copy of this one.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 891).

6.434.3.2 virtual void activemq::core::RedeliveryPolicy::configure (const decaf::util::Properties & *properties*) [virtual]

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix cms.RedeliveryPolicy.XXX where XXX is the name of a property with a public setter method. For instance cms.RedeliveryPolicy.useExponentialBackOff will be used to set the value of the use exponential back off toggle.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

Parameters

properties	The Properties object used to configure this object.

Exceptions

NumberFormatException	if a property that is numeric cannot be converted
IllegalArgumentException	if a property can't be converted to the correct type.

6.434.3.3 virtual double activemq::core::RedeliveryPolicy::getBackOffMultiplier ( ) const [pure virtual]

Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 891).

6.434.3.4 virtual short activemq::core::RedeliveryPolicy::getCollisionAvoidancePercent ( ) const [pure virtual]

Returns

the currently set Collision Avoidance percentage.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 892).

6.434.3.5 virtual long long activemq::core::RedeliveryPolicy::getInitialRedeliveryDelay ( ) const [pure virtual]

Gets the initial time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed initially.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 892).

6.434.3.6 virtual int activemq::core::RedeliveryPolicy::getMaximumRedeliveries ( ) const [pure virtual]

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

Returns

maximum allowed redeliveries for a message.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 892).

6.434.3.7 virtual long long activemq::core::RedeliveryPolicy::getNextRedeliveryDelay (long long *previousDelay*) [pure virtual]

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

Parameters

previousDelay	The last delay that was used between message redeliveries.

Returns

the new delay to use before attempting another redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 892).

6.434.3.8 virtual long long activemq::core::RedeliveryPolicy::getRedeliveryDelay ( ) const [pure virtual]

Gets the time that redelivery of messages is delayed.

Returns

the time in milliseconds that redelivery is delayed.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 892).

6.434.3.9 virtual bool activemq::core::RedeliveryPolicy::isUseCollisionAvoidance ( ) const [pure virtual]

Returns

whether or not collision avoidance is enabled for this Policy.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 893).

6.434.3.10 virtual bool activemq::core::RedeliveryPolicy::isUseExponentialBackOff ( ) const [pure virtual]

Returns

whether or not the exponential back off option is enabled.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 893).

6.434.3.11 virtual void activemq::core::RedeliveryPolicy::setBackOffMultiplier (double *value*) [pure virtual]

Sets the Back-Off Multiplier for Message Redelivery.

Parameters

value	The new value for the back-off multiplier.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 893).

6.434.3.12 virtual void activemq::core::RedeliveryPolicy::setCollisionAvoidancePercent (short *value*) [pure virtual]

Parameters

value	The collision avoidance percentage setting.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 893).

6.434.3.13 virtual void activemq::core::RedeliveryPolicy::setInitialRedeliveryDelay (long long *value*) [pure virtual]

Sets the initial time that redelivery will be delayed.

Parameters

value	Time in Milliseconds to wait before starting redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 893).

6.434.3.14 virtual void activemq::core::RedeliveryPolicy::setMaximumRedeliveries (int *maximumRedeliveries*) [pure virtual]

Sets the Maximum allowable redeliveries for a Message.

Parameters

maximum-Redeliveries	The maximum number of times that a message will be redelivered.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 893).

6.434.3.15 virtual void activemq::core::RedeliveryPolicy::setRedeliveryDelay (long long *value*) [pure virtual]

Sets the time that redelivery will be delayed.

Parameters

value	Time in Milliseconds to wait before the next redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 894).

6.434.3.16 virtual void activemq::core::RedeliveryPolicy::setUseCollisionAvoidance (bool *value*) [pure virtual]

Parameters

value	Enable or Disable collision avoidance for this Policy.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 894).

6.434.3.17 virtual void activemq::core::RedeliveryPolicy::setUseExponentialBackOff (bool *value*) [pure virtual]

Parameters

value	Enable or Disable the exponential back off multiplier option.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 894).

6.434.4 Field Documentation

6.434.4.1 const long long activemq::core::RedeliveryPolicy::NO_MAXIMUM_REDELIVERIES [static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**RedeliveryPolicy.h**

6.435 decaf::util::concurrent::locks::ReentrantLock Class Reference

A reentrant mutual exclusion **Lock** (p. 1305) 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** ()

  Acquires the lock.
- virtual void **lockInterruptibly** ()

  Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** ()

  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)

  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** ()

  Attempts to release this lock.
- virtual **Condition** * **newCondition** ()

  *Returns a **Condition** (p. 715) instance for use with this **Lock** (p. 1305) 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.

6.435.1 Detailed Description

A reentrant mutual exclusion **Lock** (p. 1305) with extended capabilities.

A **ReentrantLock** (p. 1749) 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 **isHeldBy-CurrentThread()** (p. 1751), and **getHoldCount()** (p. 1750).

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 lock;
// ...
```

public:

```
void m() {
    lock.lock();  // block until condition holds

    try {
        // ... method body
    } finally {
        lock.unlock()
    }
}
```

}

In addition to implementing the **Lock** (p. 1305) interface, this class defines methods isLocked and getLockQueue-Length, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

Since

1.0

6.435.2 Constructor & Destructor Documentation

6.435.2.1 decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ( )

6.435.2.2 virtual decaf::util::concurrent::locks::ReentrantLock::~ReentrantLock ( ) [virtual]

6.435.3 Member Function Documentation

6.435.3.1 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 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

6.435.3.2 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

6.435.3.3 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. 1749) lock; // ...

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. 1749) lock; // ...

public: void m() { assert !lock.**isHeldByCurrentThread()** (p. 1751); lock.lock(); try { // ... method body } finally { lock.unlock(); } } }

Returns

true if current thread holds this lock and false otherwise

6.435.3.4 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

6.435.3.5 virtual void decaf::util::concurrent::locks::ReentrantLock::lock ( ) [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. 1306).

6.435.3.6 virtual void decaf::util::concurrent::locks::ReentrantLock::lockInterruptibly ( ) [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. 1307).

6.435.3.7 virtual Condition* decaf::util::concurrent::locks::ReentrantLock::newCondition ( ) [virtual]

Returns a **Condition** (p. 715) instance for use with this **Lock** (p. 1305) instance.

The returned **Condition** (p. 715) instance supports the same usages as do the **Mutex** (p. 1519) Class' methods (wait, notify, and notifyAll).

- If this lock is not held when any of the **Condition** (p. 715) 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. 715).
UnsupportedOperationException	if this **Lock** (p. 1305) implementation does not support conditions

Implements **decaf::util::concurrent::locks::Lock** (p. 1307).

6.435.3.8 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

6.435.3.9 virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ( ) [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. 1753) 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. 2141)) 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. 1308).

6.435.3.10 virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock (long long *time,* const TimeUnit & *unit*) [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. 1753) 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. 1308).

6.435.3.11 virtual void decaf::util::concurrent::locks::ReentrantLock::unlock ( ) [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. 1309).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**ReentrantLock.h**

6.436 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 ()

6.436.1 Constructor & Destructor Documentation

6.436.1.1 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( ) throw () [inline]

Default Constructor.

6.436.1.2 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

6.436.1.3 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const RejectedExecutionException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	- The Exception to copy in this new instance.

6.436.1.4 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.

6.436.1.5 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

6.436.1.6 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

6.436.1.7 virtual decaf::util::concurrent::RejectedExecutionException::~RejectedExecutionException ( ) throw () [inline, virtual]

6.436.2 Member Function Documentation

6.436.2.1 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. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**RejectedExecutionException.h**

6.437 decaf::util::concurrent::RejectedExecutionHandler Class Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. 2104).

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

Inheritance diagram for decaf::util::concurrent::RejectedExecutionHandler:

Public Member Functions

- **RejectedExecutionHandler** ()
- virtual ~**RejectedExecutionHandler** ()
- virtual void **rejectedExecution** (**decaf::lang::Runnable** *r, **ThreadPoolExecutor** *executer)=0

  *Method that may be invoked by a **ThreadPoolExecutor** (p. 2104) when **execute** (p. 2110) cannot accept a task.*

6.437.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. 2104).

Since

1.0

6.437.2 Constructor & Destructor Documentation

6.437.2.1 decaf::util::concurrent::RejectedExecutionHandler::RejectedExecutionHandler ( )

6.437.2.2 virtual decaf::util::concurrent::RejectedExecutionHandler::~RejectedExecutionHandler ( ) [virtual]

6.437.3 Member Function Documentation

6.437.3.1 virtual void decaf::util::concurrent::RejectedExecutionHandler::rejectedExecution (decaf::lang::Runnable * *r*, ThreadPoolExecutor * *executer*) [pure virtual]

Method that may be invoked by a **ThreadPoolExecutor** (p. 2104) when **execute** (p. 2110) cannot accept a task.

This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p. 1004).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p. 1755), which will be propagated to the caller of **execute** (p. 2110).

Parameters

r	The pointer to the runnable task requested to be executed.
executor	The pointer to the executor attempting to execute this task.

Exceptions

RejectedExecution-Exception *(p. 1755)*	if there is no remedy.

Implemented in **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy** (p. 948).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**RejectedExecutionHandler.h**

6.438 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **RemoveInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual **const Pointer**
  < **DataStructure** > & **getObjectId** () **const**
- virtual **Pointer**< **DataStructure** > & **getObjectId** ()
- virtual void **setObjectId** (**const Pointer**< **DataStructure** > &**objectId**)
- virtual long long **getLastDeliveredSequenceId** () **const**
- virtual void **setLastDeliveredSequenceId** (long long **lastDeliveredSequenceId**)
- virtual bool **isRemoveInfo** () **const**
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ∗visitor)

  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 Attributes

- **Pointer**< **DataStructure** > **objectId**
- long long **lastDeliveredSequenceId**

6.438.1 Constructor & Destructor Documentation

6.438.1.1 activemq::commands::RemoveInfo::RemoveInfo ( )

6.438.1.2 virtual activemq::commands::RemoveInfo::∼RemoveInfo ( ) [virtual]

6.438.2 Member Function Documentation

6.438.2.1 virtual RemoveInfo∗ activemq::commands::RemoveInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.438.2.2 virtual void activemq::commands::RemoveInfo::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. 382).

6.438.2.3 virtual bool activemq::commands::RemoveInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.438.2.4 virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.438.2.5 virtual long long activemq::commands::RemoveInfo::getLastDeliveredSequenceId ( ) const [virtual]

6.438.2.6 virtual const Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId ( ) const [virtual]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.438.2.7 virtual Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId ( ) [virtual]

6.438.2.8 virtual bool activemq::commands::RemoveInfo::isRemoveInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isRemoveInfo()** (p. 1760) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 384).

6.438.2.9 virtual void activemq::commands::RemoveInfo::setLastDeliveredSequenceId (long long *lastDeliveredSequenceId*) [virtual]

6.438.2.10 virtual void activemq::commands::RemoveInfo::setObjectId (const Pointer< DataStructure > & *objectId*) [virtual]

6.438.2.11 virtual std::string activemq::commands::RemoveInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.438.2.12 virtual Pointer<Command> activemq::commands::RemoveInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.438.3 Field Documentation

6.438.3.1 const unsigned char activemq::commands::RemoveInfo::ID_REMOVEINFO = 12 [static]

6.438.3.2 long long activemq::commands::RemoveInfo::lastDeliveredSequenceId [protected]

6.438.3.3 Pointer<DataStructure> activemq::commands::RemoveInfo::objectId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**RemoveInfo.h**

6.439 activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 1761).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/RemoveInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.439.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 1761).

NOTE!: This file is 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::generated::RemoveInfoMarshaller::RemoveInfoMarshaller ( ) [inline]

6.439.2.2 virtual activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::∼RemoveInfoMarshaller ( ) [inline, virtual]

6.439.3 Member Function Documentation

6.439.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.439.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.439.3.3 virtual void activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::looseMarshal (OpenWireFormat ∗ *format*, commands::DataStructure ∗ *command*, decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.439.3.4 virtual void activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.439.3.5 virtual int activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.439.3.6 virtual void activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.439.3.7 virtual void activemq::wireformat::openwire::marshal::generated::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**RemoveInfoMarshaller.h**

6.440 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **RemoveSubscriptionInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 & **getSubcriptionName** () **const**
- virtual std::string & **getSubcriptionName** ()
- virtual void **setSubcriptionName** (**const** std::string &**subcriptionName**)
- 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)

  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 Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **subcriptionName**
- std::string **clientId**

6.440.1 Constructor & Destructor Documentation

6.440.1.1 activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo ( )

6.440.1.2 virtual activemq::commands::RemoveSubscriptionInfo::~RemoveSubscriptionInfo ( ) [virtual]

6.440.2 Member Function Documentation

6.440.2.1 virtual RemoveSubscriptionInfo* activemq::commands::RemoveSubscriptionInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.440.2.2 virtual void activemq::commands::RemoveSubscriptionInfo::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. 382).

6.440.2.3 virtual bool activemq::commands::RemoveSubscriptionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.440.2.4 virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId ( ) const [virtual]

6.440.2.5 virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId ( ) [virtual]

6.440.2.6 virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId ( ) const [virtual]

6.440.2.7 virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId ( ) [virtual]

6.440.2.8 virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.440.2.9 virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubcriptionName ( ) const [virtual]

6.440.2.10 virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubcriptionName ( ) [virtual]

6.440.2.11 virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isRemoveSubscriptionInfo()** (p. 1766) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 385).

6.440.2.12 virtual void activemq::commands::RemoveSubscriptionInfo::setClientId (const std::string & *clientId*) [virtual]

6.440.2.13 virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.440.2.14 virtual void activemq::commands::RemoveSubscriptionInfo::setSubcriptionName (const std::string & *subcriptionName*) [virtual]

6.440.2.15 virtual std::string activemq::commands::RemoveSubscriptionInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.440.2.16 virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.440.3 Field Documentation

6.440.3.1 std::string activemq::commands::RemoveSubscriptionInfo::clientId [protected]

6.440.3.2 Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId [protected]

6.440.3.3 const unsigned char activemq::commands::RemoveSubscriptionInfo::ID_REMOVESUBSCRIPTIONINFO = 9 [static]

6.440.3.4 std::string activemq::commands::RemoveSubscriptionInfo::subcriptionName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**RemoveSubscriptionInfo.h**

6.441 activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 1767).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/RemoveSubscriptionInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual ~**RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.441.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 1767).

NOTE!: This file is 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::generated::RemoveSubscriptionInfoMarshaller::-RemoveSubscriptionInfoMarshaller ( ) [inline]

6.441.2.2 virtual activemq::wireformat::openwire::marshal::generated::RemoveSubscription-InfoMarshaller::~RemoveSubscriptionInfoMarshaller ( ) [inline, virtual]

6.441.3 Member Function Documentation

6.441.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal-::generated::RemoveSubscriptionInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.441.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.441.3.3 virtual void activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.441.3.4 virtual void activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.441.3.5 virtual int activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.441.3.6 virtual void activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.441.3.7 virtual void activemq::wireformat::openwire::marshal::generated::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**RemoveSubscriptionInfoMarshaller.h**

6.442 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ReplayCommand** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- int **firstNakNumber**
- int **lastNakNumber**

6.442.1 Constructor & Destructor Documentation

6.442.1.1 activemq::commands::ReplayCommand::ReplayCommand ( )

6.442.1.2 virtual activemq::commands::ReplayCommand::~ReplayCommand ( ) [virtual]

6.442.2 Member Function Documentation

6.442.2.1 virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.442.2.2 virtual void activemq::commands::ReplayCommand::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. 382).

6.442.2.3 virtual bool activemq::commands::ReplayCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.442.2.4 virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.442.2.5 virtual int activemq::commands::ReplayCommand::getFirstNakNumber ( ) const [virtual]

6.442.2.6 virtual int activemq::commands::ReplayCommand::getLastNakNumber ( ) const [virtual]

6.442.2.7 virtual void activemq::commands::ReplayCommand::setFirstNakNumber (int *firstNakNumber*) [virtual]

6.442.2.8 virtual void activemq::commands::ReplayCommand::setLastNakNumber (int *lastNakNumber*) [virtual]

6.442.2.9 virtual std::string activemq::commands::ReplayCommand::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.442.2.10 virtual Pointer<Command> activemq::commands::ReplayCommand::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.442.3 Field Documentation

6.442.3.1 int activemq::commands::ReplayCommand::firstNakNumber [protected]

6.442.3.2 const unsigned char activemq::commands::ReplayCommand::ID_REPLAYCOMMAND = 65 [static]

6.442.3.3 int activemq::commands::ReplayCommand::lastNakNumber [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ReplayCommand.h**

6.443 activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 1773).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ReplayCommandMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual ~**ReplayCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.443.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (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

6.443.2 Constructor & Destructor Documentation

6.443.2.1 activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::ReplayCommandMarshaller ( ) [inline]

6.443.2.2 virtual activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::∼ReplayCommandMarshaller ( ) [inline, virtual]

6.443.3 Member Function Documentation

6.443.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.443.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.443.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::looseMarshal (OpenWireFormat ∗ *format*, commands::DataStructure ∗ *command*, decaf::io::DataOutputStream ∗ *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.443.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.443.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.443.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.443.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ReplayCommandMarshaller.h**

6.444 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** () throw ()
- virtual **cms::Destination** * **getDestination** (**cms::Session** *session)

6.444.1 Constructor & Destructor Documentation

6.444.1.1 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor (ProducerCallback * *action*, CmsTemplate * *parent*, const std::string & *destinationName*) [inline]

6.444.1.2 virtual activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::~ResolveProducerExecutor ( ) throw () [inline, virtual]

6.444.2 Member Function Documentation

6.444.2.1 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination (cms::Session * *session*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.445 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** () throw ()
- virtual **cms::Destination** * **getDestination** (**cms::Session** *session)

6.445.1 Constructor & Destructor Documentation

6.445.1.1 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor (CmsTemplate * *parent,* const std::string & *selector,* bool *noLocal,* const std::string & *destinationName*) [inline]

6.445.1.2 virtual activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::~ResolveReceiveExecutor ( ) throw () [inline, virtual]

6.445.2 Member Function Documentation

6.445.2.1 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination (cms::Session * *session*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.446 decaf::internal::util::Resource Class Reference

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

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

Inheritance diagram for decaf::internal::util::Resource:

Public Member Functions

- virtual ~**Resource** ()

6.446.1 Detailed Description

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

Since

1.0

6.446.2 Constructor & Destructor Documentation

6.446.2.1 virtual decaf::internal::util::Resource::~Resource ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**Resource.h**

6.447 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** ()

  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.

Protected Member Functions

- **ResourceLifecycleManager** (**const ResourceLifecycleManager** &)
- **ResourceLifecycleManager** & **operator=** (**const ResourceLifecycleManager** &)

6.447.1 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.

6.447.2 Constructor & Destructor Documentation

6.447.2.1 activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager (const ResourceLifecycleManager &) [protected]

6.447.2.2 activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager ( )

6.447.2.3 virtual activemq::cmsutil::ResourceLifecycleManager::~ResourceLifecycleManager ( ) [virtual]

Destructor - calls `destroy`

6.447.3 Member Function Documentation

6.447.3.1 void activemq::cmsutil::ResourceLifecycleManager::addConnection (cms::Connection * *connection*)

Adds a connection so that its life will be managed by this object.

Parameters

connection	the object to be managed

Exceptions

CMSException	if an error occurs while performing this operation.

6.447.3.2 void activemq::cmsutil::ResourceLifecycleManager::addDestination (cms::Destination * *dest*)

Adds a destination so that its life will be managed by this object.

Parameters

dest	the object to be managed

Exceptions

CMSException	if an error occurs while performing this operation.

6.447.3.3 void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer (cms::MessageConsumer * *consumer*)

Adds a message consumer so that its life will be managed by this object.

Parameters

consumer	the object to be managed

Exceptions

CMSException	if an error occurs while performing this operation.

6.447.3.4 void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer (cms::MessageProducer * *producer*)

Adds a message producer so that its life will be managed by this object.

Parameters

producer	the object to be managed

Exceptions

CMSException	if an error occurs while performing this operation.

6.447.3.5 void activemq::cmsutil::ResourceLifecycleManager::addSession (cms::Session * *session*)

Adds a session so that its life will be managed by this object.

Parameters

session	the object to be managed

Exceptions

CMSException	if an error occurs while performing this operation.

6.447.3.6 void activemq::cmsutil::ResourceLifecycleManager::destroy ( )

Closes and destroys the contained CMS resources.

Exceptions

cms::CMSException *(p. 640)*	thrown if an error occurs.

6.447.3.7 ResourceLifecycleManager& activemq::cmsutil::ResourceLifecycleManager::operator= (const ResourceLifecycleManager &) [protected]

6.447.3.8 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**

6.448 decaf::internal::util::ResourceLifecycleManager Class Reference

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

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual ∼**ResourceLifecycleManager** ()
- virtual void **addResource** (**Resource** ∗value)

Protected Member Functions

- virtual void **destroyResources** ()

6.448.1 Detailed Description

Since

1.0

6.448.2 Constructor & Destructor Documentation

6.448.2.1 decaf::internal::util::ResourceLifecycleManager::ResourceLifecycleManager ( )

6.448.2.2 virtual decaf::internal::util::ResourceLifecycleManager::∼ResourceLifecycleManager ( ) [virtual]

6.448.3 Member Function Documentation

6.448.3.1 virtual void decaf::internal::util::ResourceLifecycleManager::addResource (Resource ∗ *value*) [virtual]

6.448.3.2 virtual void decaf::internal::util::ResourceLifecycleManager::destroyResources ( ) [protected, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ResourceLifecycleManager.h**

6.449 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **Response** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- int **correlationId**

6.449.1 Constructor & Destructor Documentation

6.449.1.1 activemq::commands::Response::Response ( )

6.449.1.2 virtual activemq::commands::Response::∼Response ( ) [virtual]

6.449.2 Member Function Documentation

6.449.2.1 virtual Response∗ activemq::commands::Response::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 829), **activemq::commands::DataResponse** (p. 864), **activemq::commands::ExceptionResponse** (p. 997), and **activemq::commands::IntegerResponse** (p. 1176).

6.449.2.2 virtual void activemq::commands::Response::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. 382).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 829), **activemq::commands::DataResponse** (p. 864), **activemq::commands::ExceptionResponse** (p. 997), and **activemq::commands::IntegerResponse** (p. 1176).

6.449.2.3 virtual bool activemq::commands::Response::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 864), **activemq::commands::ExceptionResponse** (p. 998), and **activemq::commands::IntegerResponse** (p. 1176).

6.449.2.4 virtual int activemq::commands::Response::getCorrelationId ( ) const [virtual]

6.449.2.5 virtual unsigned char activemq::commands::Response::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 864), **activemq::commands::ExceptionResponse** (p. 998), and **activemq::commands::IntegerResponse** (p. 1176).

6.449.2.6 virtual bool activemq::commands::Response::isResponse ( ) const [inline, virtual]

Returns

an answer of true to the **isResponse()** (p. 1783) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 385).

6.449.2.7 virtual void activemq::commands::Response::setCorrelationId (int *correlationId*) [virtual]

6.449.2.8 virtual std::string activemq::commands::Response::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 830), **activemq::commands::DataResponse** (p. 865), **activemq::commands::ExceptionResponse** (p. 998), and **activemq::commands::IntegerResponse** (p. 1176).

6.449.2.9 virtual Pointer<Command> activemq::commands::Response::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.449.3 Field Documentation

6.449.3.1 int activemq::commands::Response::correlationId [protected]

6.449.3.2 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**

6.450 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::LinkedList**< **Pointer**< **Command** > > &queue)=0

  *When called the **ResponseBuilder** (p. 1784) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

6.450.1 Detailed Description

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

6.450.2 Constructor & Destructor Documentation

6.450.2.1 virtual activemq::transport::mock::ResponseBuilder::~ResponseBuilder ( ) [inline, virtual]

6.450.3 Member Function Documentation

6.450.3.1 virtual void activemq::transport::mock::ResponseBuilder::buildIncomingCommands (const Pointer< Command > & *command*, decaf::util::LinkedList< Pointer< Command > > & *queue*) [pure virtual]

When called the **ResponseBuilder** (p. 1784) 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. 1600).

6.450.3.2 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. 1600).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**ResponseBuilder.h**

6.451 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)

  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command)

  Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command, unsigned int timeout)

  Sends the given command to the broker and then waits for the response.
- virtual void **start** ()

  *Starts the **Transport*** (p. *2161), the send methods of a **Transport*** (p. *2161) will throw an exception if used before the **Transport*** (p. *2161) is started.*
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- virtual void **onCommand** (**const Pointer**< **Command** > &command)

  This is called in the context of the nested transport's reading thread.
- virtual void **onTransportException** (**Transport** *source, **const decaf::lang::Exception** &ex)

  Event handler for an exception from a command transport.

6.451.1 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

6.451.2 Constructor & Destructor Documentation

6.451.2.1 activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator (const Pointer< Transport > & *next*)

Constructor.

Parameters

next	the next transport in the chain

6.451.2.2 virtual activemq::transport::correlator::ResponseCorrelator::~ResponseCorrelator ( ) [virtual]

6.451.3 Member Function Documentation

6.451.3.1 virtual void activemq::transport::correlator::ResponseCorrelator::close ( ) [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.

Reimplemented from **activemq::transport::TransportFilter** (p. 2170).

6.451.3.2 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. 2173).

6.451.3.3 virtual void activemq::transport::correlator::ResponseCorrelator::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2173).

6.451.3.4 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 that was caught.

6.451.3.5 virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2174).

6.451.3.6 virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & *command,* unsigned int *timeout*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 2174).

6.451.3.7 virtual void activemq::transport::correlator::ResponseCorrelator::start ( ) [virtual]

Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.

Exceptions

IOException	if and error occurs while starting the **Transport** (p. 2161).

Reimplemented from **activemq::transport::TransportFilter** (p. 2175).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

6.452 activemq::wireformat::openwire::marshal::generated::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 1788).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ResponseMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual ~**ResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.452.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 1788).

NOTE!: This file is 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::generated::ResponseMarshaller::ResponseMarshaller ( ) [inline]

6.452.2.2 virtual activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::~ResponseMarshaller ( ) [inline, virtual]

6.452.3 Member Function Documentation

6.452.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1178).

6.452.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1178).

6.452.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1178).

6.452.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 832), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1000), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179).

6.452.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::tightMarshal1 (OpenWireFormat * *format,* commands::DataStructure * *command,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179).

6.452.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::tightMarshal2 (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds,* utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 867), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1179).

6.452.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ResponseMarshaller::tightUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis,* utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::DataArrayResponseMarshaller** (p. 833), **activemq::wireformat::openwire::marshal::generated::DataResponseMarshaller** (p. 868), **activemq::wireformat::openwire::marshal::generated::ExceptionResponseMarshaller** (p. 1001), and **activemq::wireformat::openwire::marshal::generated::IntegerResponseMarshaller** (p. 1180).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ResponseMarshaller.h**

6.453 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** (p. 2094) class in the context of the thread.*

6.453.1 Detailed Description

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

6.453.2 Constructor & Destructor Documentation

6.453.2.1 virtual decaf::lang::Runnable::~Runnable ( ) [inline, virtual]

6.453.3 Member Function Documentation

6.453.3.1 virtual void decaf::lang::Runnable::run ( ) [pure virtual]

Run method - called by the **Thread** (p. 2094) class in the context of the thread.

Implemented in **activemq::transport::IOTransport** (p. 1205), **decaf::lang::Thread** (p. 2100), **activemq::threads::CompositeTaskRunner** (p. 694), **activemq::threads::DedicatedTaskRunner** (p. 886), **activemq::transport::mock::InternalCommandListener** (p. 1184), **activemq::transport::inactivity::WriteChecker** (p. 2255), **activemq::transport::inactivity::ReadChecker** (p. 1734), and **activemq::threads::SchedulerTimerTask** (p. 1798).

Referenced by decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::rejectedExecution().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runnable.h**

6.454 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. 1793) 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.

6.454.1 Constructor & Destructor Documentation

6.454.1.1 virtual decaf::lang::Runtime::~Runtime ( ) [inline, virtual]

6.454.2 Member Function Documentation

6.454.2.1 static Runtime* decaf::lang::Runtime::getRuntime ( ) [static]

Gets the single instance of the Decaf **Runtime** (p. 1793) for this Process.

Returns

pointer to the single Decaf **Runtime** (p. 1793) instance that exists for this process

6.454.2.2 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.

6.454.2.3 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.

6.454.2.4 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**

6.455 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 ()

6.455.1 Constructor & Destructor Documentation

6.455.1.1 decaf::lang::exceptions::RuntimeException::RuntimeException ( ) throw () [inline]

Default Constructor.

6.455.1.2 decaf::lang::exceptions::RuntimeException::RuntimeException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other ActiveMQException.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.455.1.3 decaf::lang::exceptions::RuntimeException::RuntimeException (const RuntimeException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The **Exception** (p. 990) whose data is to be copied into this one.

6.455.1.4 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

6.455.1.5 decaf::lang::exceptions::RuntimeException::RuntimeException (const std::exception ∗ *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.455.1.6 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

6.455.1.7 virtual decaf::lang::exceptions::RuntimeException::~RuntimeException ( ) throw () [inline, virtual]

6.455.2 Member Function Documentation

6.455.2.1 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. 990) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

Reimplemented in **decaf::util::NoSuchElementException** (p. 1539), and **decaf::util::ConcurrentModificationException** (p. 701).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**RuntimeException.h**

6.456 activemq::threads::Scheduler Class Reference

Scheduler (p. 1796) class for use in executing Runnable Tasks either periodically or one time only with optional delay.

`#include <src/main/activemq/threads/Scheduler.h>`

Inheritance diagram for activemq::threads::Scheduler:

Public Member Functions

- **Scheduler** (**const** std::string &name)
- virtual ~**Scheduler** ()
- void **executePeriodically** (**decaf::lang::Runnable** *task, long long period, bool ownsTask=true)
- void **schedualPeriodically** (**decaf::lang::Runnable** *task, long long period, bool ownsTask=true)
- void **cancel** (**decaf::lang::Runnable** *task)
- void **executeAfterDelay** (**decaf::lang::Runnable** *task, long long delay, bool ownsTask=true)
- void **shutdown** ()

Protected Member Functions

- virtual void **doStart** ()

  Performs the actual start operation on the service, acquiring all the resources needed to run the service.
- virtual void **doStop** (**activemq::util::ServiceStopper** *stopper)

  Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.

6.456.1 Detailed Description

Scheduler (p. 1796) class for use in executing Runnable Tasks either periodically or one time only with optional delay.

Since

3.3.0

6.456.2 Constructor & Destructor Documentation

6.456.2.1 activemq::threads::Scheduler::Scheduler (const std::string & *name*)

6.456.2.2 virtual activemq::threads::Scheduler::~Scheduler ( ) [virtual]

6.456.3 Member Function Documentation

6.456.3.1 void activemq::threads::Scheduler::cancel (decaf::lang::Runnable * *task*)

6.456.3.2 virtual void activemq::threads::Scheduler::doStart ( ) [protected, virtual]

Performs the actual start operation on the service, acquiring all the resources needed to run the service.

Must be implemented in derived class.

Implements **activemq::util::ServiceSupport** (p. 1829).

6.456.3.3 virtual void activemq::threads::Scheduler::doStop (activemq::util::ServiceStopper * *stopper*) [protected, virtual]

Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.

Implements **activemq::util::ServiceSupport** (p. 1830).

6.456.3.4 void **activemq::threads::Scheduler::executeAfterDelay** (**decaf::lang::Runnable** * *task,* long long *delay,* bool *ownsTask* = `true`)

6.456.3.5 void **activemq::threads::Scheduler::executePeriodically** (**decaf::lang::Runnable** * *task,* long long *period,* bool *ownsTask* = `true`)

6.456.3.6 void **activemq::threads::Scheduler::schedualPeriodically** (**decaf::lang::Runnable** * *task,* long long *period,* bool *ownsTask* = `true`)

6.456.3.7 void **activemq::threads::Scheduler::shutdown** ( )

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**Scheduler.h**

6.457 activemq::threads::SchedulerTimerTask Class Reference

Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.

`#include <src/main/activemq/threads/SchedulerTimerTask.h>`

Inheritance diagram for activemq::threads::SchedulerTimerTask:

Public Member Functions

- **SchedulerTimerTask** (**decaf::lang::Runnable** *task, bool ownsTask=true)
- virtual ~**SchedulerTimerTask** ()
- virtual void **run** ()

  Run method - called by the Thread class in the context of the thread.

6.457.1 Detailed Description

Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.

Since

3.3.0

6.457.2 Constructor & Destructor Documentation

6.457.2.1 activemq::threads::SchedulerTimerTask::SchedulerTimerTask (**decaf::lang::Runnable** * *task,* bool *ownsTask* = `true`)

6.457.2.2 virtual **activemq::threads::SchedulerTimerTask::~SchedulerTimerTask** ( ) `[virtual]`

6.457.3 Member Function Documentation

6.457.3.1 virtual void **activemq::threads::SchedulerTimerTask::run** ( ) `[virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 1793).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**SchedulerTimerTask.h**

6.458 decaf::security::SecureRandom Class Reference

`#include <src/main/decaf/security/SecureRandom.h>`

Inheritance diagram for decaf::security::SecureRandom:

Public Member Functions

- **SecureRandom** ()

  Creates a new instance of a secure random number generator that implements the default random number algorithm.
- **SecureRandom** (**const** std::vector< unsigned char > &seed)

  Creates a new instance of a secure random number generator that implements the default random number algorithm.
- **SecureRandom** (**const** unsigned char *seed, int size)

  Creates a new instance of a secure random number generator that implements the default random number algorithm.
- virtual ~**SecureRandom** ()
- virtual void **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. *1729)*
- virtual void **nextBytes** (unsigned char *buf, int size)

  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. *1729)*

  Exceptions

NullPointerException	*if buff is NULL*
IllegalArgumentException	*if size is negative*

- virtual 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.

  Parameters

seed	*the seed that alters the state of the random number generator*

  See also

  next *(*p. *1729)*
  Random() *(*p. *1729)*
  #Random(long)
- virtual void **setSeed** (**const** std::vector< unsigned char > &seed)

  Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.
- virtual void **setSeed** (**const** unsigned char *seed, int size)

  Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

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.*
  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. *1730*)
  nextDouble (p. *1730*)
  nextFloat (p. *1731*)
  nextInt() (p. *1731*)
  nextInt(int) (p. *1731*)
  nextGaussian (p. *1731*)
  nextLong (p. *1732*)

6.458.1 Detailed Description

Since

1.0

6.458.2 Constructor & Destructor Documentation

6.458.2.1 decaf::security::SecureRandom::SecureRandom ( )

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 1799) instance that is created with this constructor is unseeded and can be seeded by calling the setSeed method. Calls to nextBytes on an unseeded **SecureRandom** (p. 1799) result in the object seeding itself.

6.458.2.2 decaf::security::SecureRandom::SecureRandom (const std::vector< unsigned char > & *seed*)

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 1799) instance created by this constructor is seeded using the passed byte array.

Parameters

seed	The seed bytes to use to seed this secure random number generator.

6.458.2.3 decaf::security::SecureRandom::SecureRandom (const unsigned char * *seed*, int *size*)

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 1799) instance created by this constructor is seeded using the passed byte array.

Parameters

seed	The seed bytes to use to seed this secure random number generator.
size	The number of bytes in the seed buffer.

Exceptions

NullPointerException	if the seed buffer is NULL.
IllegalArgumentException	if the size value is negative.

6.458.2.4 virtual decaf::security::SecureRandom::~SecureRandom ( ) [virtual]

6.458.3 Member Function Documentation

6.458.3.1 virtual int decaf::security::SecureRandom::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. 1730)
nextDouble (p. 1730)
nextFloat (p. 1731)
nextInt() (p. 1731)
nextInt(int) (p. 1731)
nextGaussian (p. 1731)
nextLong (p. 1732)

Reimplemented from **decaf::util::Random** (p. 1729).

6.458.3.2 virtual void decaf::security::SecureRandom::nextBytes (std::vector< unsigned char > & *buf*) [virtual]

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. 1729)

Reimplemented from **decaf::util::Random** (p. 1730).

6.458.3.3 virtual void decaf::security::SecureRandom::nextBytes (unsigned char * *buf*, int *size*) [virtual]

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. 1729)

Exceptions

NullPointerException	if buff is NULL
IllegalArgumentException	if size is negative

Reimplemented from **decaf::util::Random** (p. 1730).

6.458.3.4 virtual void decaf::security::SecureRandom::setSeed (unsigned long long *seed*) [virtual]

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. 1729)
Random() (p. 1729)
#Random(long)

Reimplemented from **decaf::util::Random** (p. 1732).

6.458.3.5 virtual void decaf::security::SecureRandom::setSeed (const std::vector< unsigned char > & *seed*) [virtual]

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Parameters

seed	A vector of bytes that is used update the seed of the RNG.

6.458.3.6 virtual void decaf::security::SecureRandom::setSeed (const unsigned char * *seed*, int *size*) [virtual]

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

Parameters

seed	The seed bytes to use to seed this secure random number generator.
size	The number of bytes in the seed buffer.

Exceptions

NullPointerException	if the seed buffer is NULL.
IllegalArgumentException	if the size value is negative.

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SecureRandom.h**

6.459 decaf::internal::security::SecureRandomImpl Class Reference

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

`#include <src/main/decaf/internal/security/unix/SecureRandomImpl.h>`

Inheritance diagram for decaf::internal::security::SecureRandomImpl:

Public Member Functions

- **SecureRandomImpl** ()
- virtual ∼**SecureRandomImpl** ()
- virtual void **providerSetSeed** (**const** unsigned char ∗seed, int size)

  Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char ∗bytes, int numBytes)

  Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.
- virtual unsigned char ∗ **providerGenerateSeed** (int numBytes)

  Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.
- **SecureRandomImpl** ()
- virtual ∼**SecureRandomImpl** ()
- virtual void **providerSetSeed** (**const** unsigned char ∗seed, int size)

  Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char ∗bytes, int numBytes)

  Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.
- virtual unsigned char ∗ **providerGenerateSeed** (int numBytes)

  Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

6.459.1 Detailed Description

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Secure Random Number Generator for Windows based platforms that attempts to obtain secure bytes with high entropy from known sources.

If the platform does not have a source of secure bytes then the platform random number generator is used if one exists otherwise the Decaf RNG is used as a last resort.

Since

1.0

6.459.2 Constructor & Destructor Documentation

6.459.2.1 decaf::internal::security::SecureRandomImpl::SecureRandomImpl ( )

6.459.2.2 virtual decaf::internal::security::SecureRandomImpl::~SecureRandomImpl ( ) [virtual]

6.459.2.3 decaf::internal::security::SecureRandomImpl::SecureRandomImpl ( )

6.459.2.4 virtual decaf::internal::security::SecureRandomImpl::~SecureRandomImpl ( ) [virtual]

6.459.3 Member Function Documentation

6.459.3.1 virtual unsigned char* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int *numBytes*) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

numBytes	The number of bytes that should be generated for the new seed array.

Implements **decaf::security::SecureRandomSpi** (p. 1806).

6.459.3.2 virtual unsigned char* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int *numBytes*) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

numBytes	The number of bytes that should be generated for the new seed array.

Implements **decaf::security::SecureRandomSpi** (p. 1806).

6.459.3.3 virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes (unsigned char * *bytes,* int *numBytes*) [virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

bytes	The array that will be filled with random bytes equal to size.
numBytes	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 1806).

6.459.3.4 virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes (unsigned char * *bytes,* int *numBytes*) [virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

bytes	The array that will be filled with random bytes equal to size.
numBytes	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 1806).

6.459.3.5 virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed (const unsigned char * *seed,* int *size*) [virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

seed	The array of bytes used to update the generators seed.
size	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 1806).

6.459.3.6 virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed (const unsigned char * *seed,* int *size*) [virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

seed	The array of bytes used to update the generators seed.
size	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 1806).

The documentation for this class was generated from the following files:

- src/main/decaf/internal/security/unix/**SecureRandomImpl.h**
- src/main/decaf/internal/security/windows/**SecureRandomImpl.h**

6.460 decaf::security::SecureRandomSpi Class Reference

Interface class used by Security Service Providers to implement a source of secure random bytes.

`#include <src/main/decaf/security/SecureRandomSpi.h>`

Inheritance diagram for decaf::security::SecureRandomSpi:

Public Member Functions

- **SecureRandomSpi** ()
- virtual ~**SecureRandomSpi** ()
- virtual void **providerSetSeed** (**const** unsigned char *seed, int size)=0

  Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.
- virtual void **providerNextBytes** (unsigned char *bytes, int numBytes)=0

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

- virtual unsigned char * **providerGenerateSeed** (int numBytes)=0

  Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

6.460.1 Detailed Description

Interface class used by Security Service Providers to implement a source of secure random bytes.

Since

1.0

6.460.2 Constructor & Destructor Documentation

6.460.2.1 decaf::security::SecureRandomSpi::SecureRandomSpi ( )

6.460.2.2 virtual decaf::security::SecureRandomSpi::~SecureRandomSpi ( ) [virtual]

6.460.3 Member Function Documentation

6.460.3.1 virtual unsigned char* decaf::security::SecureRandomSpi::providerGenerateSeed (int *numBytes*) [pure virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

Parameters

numBytes	The number of bytes that should be generated for the new seed array.

Implemented in **decaf::internal::security::SecureRandomImpl** (p. 1804), and **decaf::internal::security::SecureRandomImpl** (p. 1804).

6.460.3.2 virtual void decaf::security::SecureRandomSpi::providerNextBytes (unsigned char * *bytes,* int *numBytes*) [pure virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

Parameters

bytes	The array that will be filled with random bytes equal to size.
numBytes	The number of bytes to generate and write into the bytes array.

Implemented in **decaf::internal::security::SecureRandomImpl** (p. 1804), and **decaf::internal::security::SecureRandomImpl** (p. 1804).

6.460.3.3 virtual void decaf::security::SecureRandomSpi::providerSetSeed (const unsigned char * *seed,* int *size*) [pure virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

Parameters

seed	The array of bytes used to update the generators seed.
size	The size of the passed byte array.

Implemented in **decaf::internal::security::SecureRandomImpl** (p. 1805), and **decaf::internal::security::-SecureRandomImpl** (p. 1805).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SecureRandomSpi.h**

6.461 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. 1807) with the given number of permits and nonfair fairness setting.*
- **Semaphore** (int permits, bool fair)

  *Creates a **Semaphore** (p. 1807) with the given number of permits and the given fairness setting.*
- virtual ∼**Semaphore** ()
- void **acquire** ()

  Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.
- void **acquireUninterruptibly** ()

  Acquires a permit from this semaphore, blocking until one is available.
- bool **tryAcquire** ()

  Acquires a permit from this semaphore, only if one is available at the time of invocation.
- bool **tryAcquire** (long long timeout, **const TimeUnit** &unit)

  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** ()

  Releases a permit, returning it to the semaphore.
- void **acquire** (int permits)

  Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.
- void **acquireUninterruptibly** (int permits)

  Acquires the given number of permits from this semaphore, blocking until all are available.
- bool **tryAcquire** (int permits)

  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)

  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)

  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** ()

  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.

6.461.1 Detailed Description

A counting semaphore.

Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 1809) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 1811) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 1807) 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. 1807) available;

std::vector<std::string> items; std::vector<bool> used;

Mutex (p. 1519) lock;

public:

Pool() : available(MAX_AVAILABLE, true) { used.resize(MAX_AVAILABLE); items.resize(MAX_AVAILABLE); }

std::string getItem() throws InterruptedException { available.acquire(); return getNextAvailableItem(); }

void putItem(std::string x) { if(markAsUnused(x)) { available.release(); } }

std::string getNextAvailableItem() {

synchronized(&lock) (p. 2533) { 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. 2533) { 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. 1809) 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. 1304) 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. 1809) 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

6.461.2 Constructor & Destructor Documentation

6.461.2.1 decaf::util::concurrent::Semaphore::Semaphore (int *permits*)

Creates a **Semaphore** (p. 1807) 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.

6.461.2.2 decaf::util::concurrent::Semaphore::Semaphore (int *permits,* bool *fair*)

Creates a **Semaphore** (p. 1807) 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

6.461.2.3 virtual decaf::util::concurrent::Semaphore::~Semaphore ( ) [virtual]

6.461.3 Member Function Documentation

6.461.3.1 void decaf::util::concurrent::Semaphore::acquire ( )

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. 1811) 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. 1807).

6.461.3.2 void decaf::util::concurrent::Semaphore::acquire (int *permits*)

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. 1811).

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. 1807).

6.461.3.3 void decaf::util::concurrent::Semaphore::acquireUninterruptibly ( )

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. 1811) 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. 1807).

6.461.3.4 void decaf::util::concurrent::Semaphore::acquireUninterruptibly (int *permits*)

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. 1807).

6.461.3.5 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

6.461.3.6 int decaf::util::concurrent::Semaphore::drainPermits ( )

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. 1807).

6.461.3.7 bool decaf::util::concurrent::Semaphore::isFair ( ) const

Returns

true if this semaphore has fairness set true

6.461.3.8 void decaf::util::concurrent::Semaphore::release ( )

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. 1809). 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. 1807).

6.461.3.9 void decaf::util::concurrent::Semaphore::release (int *permits*)

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. 1807).

6.461.3.10 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

6.461.3.11 bool decaf::util::concurrent::Semaphore::tryAcquire ( )

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. 1812) 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. 2141)) 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. 1807).

6.461.3.12 bool decaf::util::concurrent::Semaphore::tryAcquire (long long *timeout,* const TimeUnit & *unit*)

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. 1811) 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. 1807).

6.461.3.13 bool decaf::util::concurrent::Semaphore::tryAcquire (int *permits*)

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. 2141)) 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. 1807).

6.461.3.14 bool decaf::util::concurrent::Semaphore::tryAcquire (int *permits,* long long *timeout,* const TimeUnit & *unit*)

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. 1811).

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. 1811).

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. 1807).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Semaphore.h**

6.462 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** () throw ()
- virtual void **doInCms** (**cms::Session** ∗session, **cms::MessageProducer** ∗producer)

  Execute an action given a session and producer.

6.462.1 Constructor & Destructor Documentation

6.462.1.1 activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor (MessageCreator ∗ *messageCreator,* CmsTemplate ∗ *parent*) [inline]

6.462.1.2 virtual activemq::cmsutil::CmsTemplate::SendExecutor::∼SendExecutor ( ) throw () [inline, virtual]

6.462.2 Member Function Documentation

6.462.2.1 virtual void activemq::cmsutil::CmsTemplate::SendExecutor::doInCms (cms::Session ∗ *session,* cms::MessageProducer ∗ *producer*) [inline, virtual]

Execute an action given a session and producer.

Parameters

session	the CMS `Session`
producer	the CMS `Producer`

Exceptions

cms::CMSException *(p. 640)*	if thrown by CMS API methods

Implements **activemq::cmsutil::ProducerCallback** (p. 1689).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.463 decaf::net::ServerSocket Class Reference

This class implements server sockets.

`#include <src/main/decaf/net/ServerSocket.h>`

Inheritance diagram for decaf::net::ServerSocket:

Public Member Functions

- **ServerSocket** ()

  Creates a non-bound server socket.
- **ServerSocket** (int port)

  *Creates a new **ServerSocket** (*p. *1816) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog)

  *Creates a new **ServerSocket** (*p. *1816) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog, **const InetAddress** ∗address)

  *Creates a new **ServerSocket** (*p. *1816) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- virtual ∼**ServerSocket** ()

  *Releases socket handle if **close()** (*p. *1820) hasn't been called.*
- virtual void **bind** (**const** std::string &host, int port)

  Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.
- virtual void **bind** (**const** std::string &host, int port, int backlog)

  Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.
- virtual **Socket** ∗ **accept** ()

  *Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (*p. *1816), the caller blocks until a connection is made.*
- virtual void **close** ()

  Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.
- virtual bool **isClosed** () **const**
- virtual bool **isBound** () **const**
- virtual int **getReceiveBufferSize** () **const**

  Gets the receive buffer size for this socket, SO_RCVBUF.
- virtual void **setReceiveBufferSize** (int size)

  Sets the receive buffer size for this socket, SO_RCVBUF.
- virtual bool **getReuseAddress** () **const**

  Gets the reuse address flag, SO_REUSEADDR.
- virtual void **setReuseAddress** (bool reuse)

  Sets the reuse address flag, SO_REUSEADDR.
- virtual int **getSoTimeout** () **const**

  Gets the timeout for socket operations, SO_TIMEOUT.
- virtual void **setSoTimeout** (int timeout)

  Sets the timeout for socket operations, SO_TIMEOUT.
- virtual int **getLocalPort** () **const**

  *Gets the port number on the Local machine that this **ServerSocket** (*p. *1816) is bound to.*
- virtual std::string **toString** () **const**

Static Public Member Functions

- static void **setSocketImplFactory** (**SocketImplFactory** *factory)

  *Sets the instance of a **SocketImplFactory** (p. 1928) that the **ServerSocket** (p. 1816) class should use when new instances of this class are created.*

Protected Member Functions

- **ServerSocket** (**SocketImpl** *impl)

  *Creates a **ServerSocket** (p. 1816) wrapping the provided **SocketImpl** (p. 1921) instance, this **Socket** (p. 1900) is considered unconnected.*
- virtual void **implAccept** (**Socket** *socket)

  *Virtual method that allows a **ServerSocket** (p. 1816) subclass to override the accept call and provide its own **SocketImpl** (p. 1921) for the socket.*
- virtual int **getDefaultBacklog** ()

  Allows a subclass to override what is considered the default backlog.
- void **checkClosed** () **const**
- void **ensureCreated** () **const**
- void **setupSocketImpl** (int port, int backlog, **const InetAddress** *ifAddress)

6.463.1 Detailed Description

This class implements server sockets.

A server socket waits for requests to come in over the network.

The actual work of the server socket is performed by an instance of the **SocketImpl** (p. 1921) class. An application can change the socket factory that creates the socket implementation to configure itself to create sockets of a particular type.

Since

1.0

6.463.2 Constructor & Destructor Documentation

6.463.2.1 decaf::net::ServerSocket::ServerSocket ( )

Creates a non-bound server socket.

6.463.2.2 decaf::net::ServerSocket::ServerSocket (int *port*)

Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 1928) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 1921) is created.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.

Exceptions

IOException	if there is an I/O error while performing this operation.
IllegalArgumentException	if the port value is negative or greater than 65535.

6.463.2.3 decaf::net::ServerSocket::ServerSocket (int *port,* int *backlog*)

Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 1928) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 1921) is created.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The the number of incoming connection attempts to queue before connections are refused.

Exceptions

IOException	if there is an I/O error while performing this operation.
IllegalArgumentException	if the port value is negative or greater than 65535.

6.463.2.4 decaf::net::ServerSocket::ServerSocket (int *port,* int *backlog,* const InetAddress * *address*)

Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 1928) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 1921) is created.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The the number of incoming connection attempts to queue before connections are refused.
ifAddress	The IP Address to bind to on the local machine.

Exceptions

IOException	if there is an I/O error while performing this operation.
IllegalArgumentException	if the port value is negative or greater than 65535.

6.463.2.5 virtual decaf::net::ServerSocket::~ServerSocket ( ) [virtual]

Releases socket handle if **close()** (p. 1820) hasn't been called.

6.463.2.6 decaf::net::ServerSocket::ServerSocket (SocketImpl * *impl*) [protected]

Creates a **ServerSocket** (p. 1816) wrapping the provided **SocketImpl** (p. 1921) instance, this **Socket** (p. 1900) is considered unconnected.

The **ServerSocket** (p. 1816) class takes ownership of this **SocketImpl** (p. 1921) pointer and will delete it when the **Socket** (p. 1900) class is destroyed.

Parameters

impl	The **SocketImpl** (p. 1921) instance to wrap.

Exceptions

NullPointerException	if the passed **SocketImpl** (p. 1921) is Null.

6.463.3 Member Function Documentation

6.463.3.1 virtual Socket* decaf::net::ServerSocket::accept ( ) [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 1816), the caller blocks until a connection is made.

If the SO_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 1932) if the operation times out.

Returns

a new **Socket** (p. 1900) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

Exceptions

IOException	if an I/O error occurs while binding the socket.
SocketException *(p. 1915)*	if an error occurs while blocking on the accept call.
SocketTimeoutException *(p. 1932)*	if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1554).

6.463.3.2 virtual void decaf::net::ServerSocket::bind (const std::string & *host,* int *port*) [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

Parameters

host	The IP address or host name.
port	The TCP port between 1..655535.

Exceptions

IOException	if an I/O error occurs while binding the socket.
IllegalArgumentException	if the parameters are not valid.

6.463.3.3 virtual void decaf::net::ServerSocket::bind (const std::string & *host,* int *port,* int *backlog*) [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

If the backlog is greater than zero it will be used instead of the default value, otherwise the default value is used and no error is generated.

Parameters

host	The IP address or host name.
port	The TCP port between 1..655535.
backlog	The size of listen backlog.

Exceptions

IOException	if an I/O error occurs while binding the socket.
IllegalArgumentException	if the parameters are not valid.

6.463.3.4 void decaf::net::ServerSocket::checkClosed ( ) const [protected]

6.463.3.5 virtual void decaf::net::ServerSocket::close ( ) [virtual]

Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.

Exceptions

IOException	if an I/O error occurs while performing this operation.

6.463.3.6 void decaf::net::ServerSocket::ensureCreated ( ) const [protected]

6.463.3.7 virtual int decaf::net::ServerSocket::getDefaultBacklog ( ) [protected, virtual]

Allows a subclass to override what is considered the default backlog.

Returns

the default backlog for connections.

6.463.3.8 virtual int decaf::net::ServerSocket::getLocalPort ( ) const [virtual]

Gets the port number on the Local machine that this **ServerSocket** (p. 1816) is bound to.

Returns

the port number of this machine that is bound, if not bound returns -1.

6.463.3.9 virtual int decaf::net::ServerSocket::getReceiveBufferSize ( ) const [virtual]

Gets the receive buffer size for this socket, SO_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

Returns

the receive buffer size in bytes.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.463.3.10 virtual bool decaf::net::ServerSocket::getReuseAddress ( ) const [virtual]

Gets the reuse address flag, SO_REUSEADDR.

Returns

True if the address can be reused.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.463.3.11 virtual int decaf::net::ServerSocket::getSoTimeout ( ) const [virtual]

Gets the timeout for socket operations, SO_TIMEOUT.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException *(p. 1915)*	Thrown if unable to retrieve the information.

6.463.3.12 virtual void decaf::net::ServerSocket::implAccept (Socket * *socket*) [protected, virtual]

Virtual method that allows a **ServerSocket** (p. 1816) subclass to override the accept call and provide its own **SocketImpl** (p. 1921) for the socket.

Parameters

socket	The socket object whose **SocketImpl** (p. 1921) should be used for the accept call.

Exceptions

IOException	if an I/O error occurs while performing this operation.

6.463.3.13 virtual bool decaf::net::ServerSocket::isBound ( ) const [virtual]

Returns

true of the server socket is bound.

6.463.3.14 virtual bool decaf::net::ServerSocket::isClosed ( ) const [virtual]

Returns

true if the close method has been called on the **ServerSocket** (p. 1816).

6.463.3.15 virtual void decaf::net::ServerSocket::setReceiveBufferSize (int *size*) [virtual]

Sets the receive buffer size for this socket, SO_RCVBUF.

Parameters

size	Number of bytes to set the receive buffer to.

Exceptions

SocketException *(p. 1915)*	if the operation fails.
IllegalArgumentException	if the value is zero or negative.

6.463.3.16 virtual void decaf::net::ServerSocket::setReuseAddress (bool *reuse*) [virtual]

Sets the reuse address flag, SO_REUSEADDR.

Parameters

reuse	If true, sets the flag.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.463.3.17 static void decaf::net::ServerSocket::setSocketImplFactory (SocketImplFactory * *factory*) [static]

Sets the instance of a **SocketImplFactory** (p. 1928) that the **ServerSocket** (p. 1816) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

Parameters

factory	The instance of a **SocketImplFactory** (p. 1928) to use when new **SocketImpl** (p. 1921) objects are created.

Exceptions

IOException	if an I/O error occurs while performing this operation.
SocketException *(p. 1915)*	if this method has already been called with a valid factory.

6.463.3.18 virtual void decaf::net::ServerSocket::setSoTimeout (int *timeout*) [virtual]

Sets the timeout for socket operations, SO_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

Parameters

timeout	The timeout in milliseconds for socket operations.

Exceptions

SocketException *(p. 1915)*	Thrown if unable to set the information.
IllegalArgumentException	if the timeout value is negative.

6.463.3.19 void decaf::net::ServerSocket::setupSocketImpl (int *port,* int *backlog,* const InetAddress * *ifAddress*) [protected]

6.463.3.20 virtual std::string decaf::net::ServerSocket::toString ( ) const [virtual]

Returns

a string representing this **ServerSocket** (p. 1816).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocket.h**

6.464 decaf::net::ServerSocketFactory Class Reference

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

`#include <src/main/decaf/net/ServerSocketFactory.h>`

Inheritance diagram for decaf::net::ServerSocketFactory:

Public Member Functions

- virtual ~**ServerSocketFactory** ()
- virtual **ServerSocket** * **createServerSocket** ()

  *Create a new **ServerSocket** (p. 1816) that is unbound.*
- virtual **ServerSocket** * **createServerSocket** (int port)=0

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
- virtual **ServerSocket** * **createServerSocket** (int port, int backlog)=0

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*
- virtual **ServerSocket** * **createServerSocket** (int port, int backlog, **const InetAddress** *address)=0

  *Create a new **ServerSocket** (p. 1816) that is bound to the given port.*

Static Public Member Functions

- static **ServerSocketFactory** * **getDefault** ()

  *Returns the Default **ServerSocket** (p. 1816) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.*

Protected Member Functions

- **ServerSocketFactory** ()

6.464.1 Detailed Description

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

Since

1.0

6.464.2 Constructor & Destructor Documentation

6.464.2.1 decaf::net::ServerSocketFactory::ServerSocketFactory ( ) [protected]

6.464.2.2 virtual decaf::net::ServerSocketFactory::~ServerSocketFactory ( ) [virtual]

6.464.3 Member Function Documentation

6.464.3.1 virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( ) [virtual]

Create a new **ServerSocket** (p. 1816) that is unbound.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 1558), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 904), and **decaf::internal::net::DefaultServerSocketFactory** (p. 896).

6.464.3.2 virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int *port*) [pure virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 1558), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 904), and **decaf::internal::net::DefaultServerSocketFactory** (p. 896).

6.464.3.3 virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int *port,* int *backlog*) [pure virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will use the specified connection backlog setting.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 1559), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 905), and **decaf::internal::net::DefaultServerSocketFactory** (p. 896).

6.464.3.4 virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int *port,* int *backlog,* const InetAddress * *address*) [pure virtual]

Create a new **ServerSocket** (p. 1816) that is bound to the given port.

The **ServerSocket** (p. 1816) will have been configured with the defaults from the factory. The **ServerSocket** (p. 1816) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 1816) will listen on all interfaces.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The number of pending connect request the **ServerSocket** (p. 1816) can queue.
address	The address of the interface on the local machine to bind to.

Returns

new **ServerSocket** (p. 1816) pointer that is owned by the caller.

Exceptions

IOException	if the **ServerSocket** (p. 1816) cannot be created for some reason.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 1559), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 905), and **decaf::internal::net::DefaultServerSocketFactory** (p. 897).

6.464.3.5 static ServerSocketFactory* decaf::net::ServerSocketFactory::getDefault ( ) [static]

Returns the Default **ServerSocket** (p. 1816) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.

Only one default **ServerSocketFactory** (p. 1823) exists for the lifetime of the Application.

Returns

the default **ServerSocketFactory** (p. 1823) for this application.

Reimplemented in **decaf::net::ssl::SSLServerSocketFactory** (p. 1946).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocketFactory.h**

6.465 activemq::util::Service Class Reference

Base interface for all classes that run as a **Service** (p. 1826) inside the application.

`#include <src/main/activemq/util/Service.h>`

Inheritance diagram for activemq::util::Service:

Public Member Functions

- virtual ~**Service** ()
- virtual void **start** ()=0
- virtual void **stop** ()=0

6.465.1 Detailed Description

Base interface for all classes that run as a **Service** (p. 1826) inside the application.

Since

3.3.0

6.465.2 Constructor & Destructor Documentation

6.465.2.1 virtual activemq::util::Service::~Service ( ) [virtual]

6.465.3 Member Function Documentation

6.465.3.1 virtual void activemq::util::Service::start ( ) [pure virtual]

Implemented in **activemq::util::ServiceSupport** (p. 1830).

6.465.3.2 virtual void activemq::util::Service::stop ( ) [pure virtual]

Implemented in **activemq::util::ServiceSupport** (p. 1830).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**Service.h**

6.466 activemq::util::ServiceListener Class Reference

Listener interface for observers of **Service** (p. 1826) related events.

`#include <src/main/activemq/util/ServiceListener.h>`

Public Member Functions

- virtual ~**ServiceListener** ()
- virtual void **started** (**const Service** *target)=0

  indicates that the target service has completed its start operation.
- virtual void **stopped** (**const Service** *target)=0

  indicates that the target service has completed its stop operation.

6.466.1 Detailed Description

Listener interface for observers of **Service** (p. 1826) related events.

Since

3.3.0

6.466.2 Constructor & Destructor Documentation

6.466.2.1 virtual activemq::util::ServiceListener::~ServiceListener ( ) [virtual]

6.466.3 Member Function Documentation

6.466.3.1 virtual void activemq::util::ServiceListener::started (const Service * *target*) [pure virtual]

indicates that the target service has completed its start operation.

Parameters

target	The service that triggered this notification.

6.466.3.2 virtual void activemq::util::ServiceListener::stopped (const Service * *target*) [pure virtual]

indicates that the target service has completed its stop operation.

Parameters

target	The service that triggered this notification.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**ServiceListener.h**

6.467 activemq::util::ServiceStopper Class Reference

`#include <src/main/activemq/util/ServiceStopper.h>`

Public Member Functions

- **ServiceStopper** ()
- virtual ∼**ServiceStopper** ()
- void **stop** (**Service** ∗service)
- void **throwFirstException** ()
- virtual void **onException** (**Service** ∗service, **decaf::lang::Exception** &ex)

6.467.1 Constructor & Destructor Documentation

6.467.1.1 activemq::util::ServiceStopper::ServiceStopper ( )

6.467.1.2 virtual activemq::util::ServiceStopper::∼ServiceStopper ( ) [virtual]

6.467.2 Member Function Documentation

6.467.2.1 virtual void activemq::util::ServiceStopper::onException (Service ∗ *service,* decaf::lang::Exception & *ex*) [virtual]

6.467.2.2 void activemq::util::ServiceStopper::stop (Service ∗ *service*)

6.467.2.3 void activemq::util::ServiceStopper::throwFirstException ( )

The documentation for this class was generated from the following file:

- src/main/activemq/util/**ServiceStopper.h**

6.468 activemq::util::ServiceSupport Class Reference

Provides a base class for **Service** (p. 1826) implementations.

`#include <src/main/activemq/util/ServiceSupport.h>`

Inheritance diagram for activemq::util::ServiceSupport:

Public Member Functions

- **ServiceSupport** (**const ServiceSupport** &)
- **ServiceSupport** & **operator=** (**const ServiceSupport** &)
- **ServiceSupport** ()
- virtual ∼**ServiceSupport** ()
- void **start** ()

  *Starts the **Service** (p. 1826), notifying any registered listeners of the start if it is successful.*
- void **stop** ()

  *Stops the **Service** (p. 1826).*
- bool **isStarted** () **const**
- bool **isStopping** () **const**
- bool **isStopped** () **const**
- void **addServiceListener** (**ServiceListener** ∗listener)

  *Adds the given listener to this **Service** (p. 1826)'s list of listeners, call retains ownership of the pointer.*
- void **removeServiceListener** (**ServiceListener** ∗llistener)

  *Removes the given listener to this **Service** (p. 1826)'s list of listeners, call retains ownership of the pointer.*

Static Public Member Functions

- static void **dispose** (**Service** ∗service)

  Safely shuts down a service.

Protected Member Functions

- virtual void **doStop** (**ServiceStopper** ∗stopper)=0

  Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.
- virtual void **doStart** ()=0

  Performs the actual start operation on the service, acquiring all the resources needed to run the service.

6.468.1 Detailed Description

Provides a base class for **Service** (p. 1826) implementations.

Since

3.3.0

6.468.2 Constructor & Destructor Documentation

6.468.2.1 activemq::util::ServiceSupport::ServiceSupport (const ServiceSupport &)

6.468.2.2 activemq::util::ServiceSupport::ServiceSupport ( )

6.468.2.3 virtual activemq::util::ServiceSupport::∼ServiceSupport ( ) [virtual]

6.468.3 Member Function Documentation

6.468.3.1 void activemq::util::ServiceSupport::addServiceListener (ServiceListener ∗ *listener*)

Adds the given listener to this **Service** (p. 1826)'s list of listeners, call retains ownership of the pointer.

6.468.3.2 static void activemq::util::ServiceSupport::dispose (Service ∗ *service*) [static]

Safely shuts down a service.

Parameters

service	The service to stop.

6.468.3.3 virtual void activemq::util::ServiceSupport::doStart ( ) [protected, pure virtual]

Performs the actual start operation on the service, acquiring all the resources needed to run the service.

Must be implemented in derived class.

Implemented in **activemq::threads::Scheduler** (p. 1797).

6.468.3.4 virtual void activemq::util::ServiceSupport::doStop (ServiceStopper * *stopper*) [protected, pure virtual]

Performs the actual stop operation on the service, ensuring that all resources held are released, must be implemented in derived class.

Implemented in **activemq::threads::Scheduler** (p. 1797).

6.468.3.5 bool activemq::util::ServiceSupport::isStarted ( ) const

Returns

true if this service has been started

6.468.3.6 bool activemq::util::ServiceSupport::isStopped ( ) const

Returns

true if this service is closed

6.468.3.7 bool activemq::util::ServiceSupport::isStopping ( ) const

Returns

true if this service is in the process of closing

6.468.3.8 ServiceSupport& activemq::util::ServiceSupport::operator= (const ServiceSupport &)

6.468.3.9 void activemq::util::ServiceSupport::removeServiceListener (ServiceListener * *llistener*)

Removes the given listener to this **Service** (p. 1826)'s list of listeners, call retains ownership of the pointer.

6.468.3.10 void activemq::util::ServiceSupport::start ( ) [virtual]

Starts the **Service** (p. 1826), notifying any registered listeners of the start if it is successful.

Implements **activemq::util::Service** (p. 1826).

6.468.3.11 void activemq::util::ServiceSupport::stop ( ) [virtual]

Stops the **Service** (p. 1826).

Implements **activemq::util::Service** (p. 1826).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**ServiceSupport.h**

6.469 cms::Session Class Reference

A **Session** (p. 1830) 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_TRANSACTED**,
  INDIVIDUAL_ACKNOWLEDGE }

Public Member Functions

- virtual ~**Session** () throw ()
- virtual void **close** ()=0

  Closes this session as well as any active child consumers or producers.
- virtual void **commit** ()=0

  Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()=0

  Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** ()=0

  Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **MessageConsumer** * **createConsumer** (**const Destination** *destination)=0

  *Creates a **MessageConsumer*** (p. *1455) for the specified destination.*
- virtual **MessageConsumer** * **createConsumer** (**const Destination** *destination, **const** std::string &selector)=0

  *Creates a **MessageConsumer*** (p. *1455) for the specified destination, using a message selector.*
- virtual **MessageConsumer** * **createConsumer** (**const Destination** *destination, **const** std::string &selector, bool noLocal)=0

  *Creates a **MessageConsumer*** (p. *1455) 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

  *Creates a durable subscriber to the specified topic, using a **Message*** (p. *1426) selector.*
- virtual **MessageProducer** * **createProducer** (**const Destination** *destination=NULL)=0

  *Creates a **MessageProducer*** (p. *1491) to send messages to the specified destination.*
- virtual **QueueBrowser** * **createBrowser** (**const cms::Queue** *queue)=0

  *Creates a new **QueueBrowser*** (p. *1726) to peek at Messages on the given **Queue*** (p. *1722).*
- virtual **QueueBrowser** * **createBrowser** (**const cms::Queue** *queue, **const** std::string &selector)=0

  *Creates a new **QueueBrowser*** (p. *1726) to peek at Messages on the given **Queue*** (p. *1722).*
- virtual **Queue** * **createQueue** (**const** std::string &queueName)=0

  *Creates a queue identity given a **Queue*** (p. *1722) name.*
- virtual **Topic** * **createTopic** (**const** std::string &topicName)=0

  *Creates a topic identity given a **Queue*** (p. *1722) name.*
- virtual **TemporaryQueue** * **createTemporaryQueue** ()=0

  *Creates a **TemporaryQueue*** (p. *2090) object.*
- virtual **TemporaryTopic** * **createTemporaryTopic** ()=0

  *Creates a **TemporaryTopic*** (p. *2091) object.*
- virtual **Message** * **createMessage** ()=0

  *Creates a new **Message*** (p. *1426).*
- virtual **BytesMessage** * **createBytesMessage** ()=0

  *Creates a **BytesMessage*** (p. *557).*

- virtual **BytesMessage** * **createBytesMessage** (**const** unsigned char *bytes, int bytesSize)=0

  *Creates a **BytesMessage** (p. 557) and sets the payload to the passed value.*
- virtual **StreamMessage** * **createStreamMessage** ()=0

  *Creates a new **StreamMessage** (p. 2020).*
- virtual **TextMessage** * **createTextMessage** ()=0

  *Creates a new **TextMessage** (p. 2093).*
- virtual **TextMessage** * **createTextMessage** (**const** std::string &text)=0

  *Creates a new **TextMessage** (p. 2093) and set the text to the value given.*
- virtual **MapMessage** * **createMapMessage** ()=0

  *Creates a new **MapMessage** (p. 1379).*
- virtual **AcknowledgeMode getAcknowledgeMode** () **const** =0

  Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () **const** =0

  *Gets if the Sessions is a Transacted **Session** (p. 1830).*
- virtual void **unsubscribe** (**const** std::string &name)=0

  Unsubscribes a durable subscription that has been created by a client.

6.469.1 Detailed Description

A **Session** (p. 1830) 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. 1722) or **Topic** (p. 2141) 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. 1455) until a message arrives. The thread may then use one or more of the **Session** (p. 1830)'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. 1830) method that can be called concurrently.
  - Invoking any other **Session** (p. 1830) method on a closed session must throw an **IllegalStateException** (p. 1098). Closing a closed session must not throw any exceptions.

Transacted Sessions

When a **Session** (p. 1830) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 1830) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 1830). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 1491) 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. 1455) 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. 1426) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 1426).

While the **Session** (p. 1830) interface implements the **Startable** (p. 1963) and **Stoppable** (p. 2016) interfaces it is not required to implement these methods and can throw an UnsupportedOperation exception if they are not available for the given CMS provider.

Since

1.0

6.469.2 Member Enumeration Documentation

6.469.2.1 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. 1426) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

6.469.3 Constructor & Destructor Documentation

6.469.3.1 virtual cms::Session::~Session ( ) throw () [virtual]

6.469.4 Member Function Documentation

6.469.4.1 virtual void cms::Session::close ( ) [pure virtual]

Closes this session as well as any active child consumers or producers.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implements **cms::Closeable** (p. 633).

Implemented in **activemq::core::ActiveMQSession** (p. 262), and **activemq::cmsutil::PooledSession** (p. 1623).

6.469.4.2 virtual void cms::Session::commit ( ) [pure virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
IllegalStateException *(p. 1098)*	- if the method is not called by a transacted session.

Implemented in **activemq::core::ActiveMQSession** (p. 262), **activemq::cmsutil::PooledSession** (p. 1623), and **activemq::core::ActiveMQXASession** (p. 339).

Referenced by activemq::cmsutil::PooledSession::commit().

6.469.4.3 virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * *queue*) [pure virtual]

Creates a new **QueueBrowser** (p. 1726) to peek at Messages on the given **Queue** (p. 1722).

Parameters

queue	the **Queue** (p. 1722) to browse

Returns

New **QueueBrowser** (p. 1726) that is owned by the caller.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
InvalidDestination-Exception *(p. 1190)*	- if the destination given is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 263), and **activemq::cmsutil::PooledSession** (p. 1623).

6.469.4.4 virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * *queue,* const std::string & *selector*) [pure virtual]

Creates a new **QueueBrowser** (p. 1726) to peek at Messages on the given **Queue** (p. 1722).

Parameters

queue	the **Queue** (p. 1722) to browse
selector	the **Message** (p. 1426) selector to filter which messages are browsed.

Returns

New **QueueBrowser** (p. 1726) that is owned by the caller.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
InvalidDestination-Exception *(p. 1190)*	- if the destination given is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 263), and **activemq::cmsutil::PooledSession** (p. 1624).

6.469.4.5 virtual BytesMessage* cms::Session::createBytesMessage ( ) [pure virtual]

Creates a **BytesMessage** (p. 557).

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 263), and **activemq::cmsutil::PooledSession** (p. 1624).

Referenced by activemq::cmsutil::PooledSession::createBytesMessage().

6.469.4.6 virtual BytesMessage* cms::Session::createBytesMessage (const unsigned char * *bytes*, int *bytesSize*) [pure virtual]

Creates a **BytesMessage** (p. 557) 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. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 264), and **activemq::cmsutil::PooledSession** (p. 1624).

6.469.4.7 virtual MessageConsumer* cms::Session::createConsumer (const Destination * *destination*) [pure virtual]

Creates a **MessageConsumer** (p. 1455) for the specified destination.

Parameters

destination	the **Destination** (p. 936) that this consumer receiving messages for.

Returns

pointer to a new **MessageConsumer** (p. 1455) that is owned by the caller (caller deletes)

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
InvalidDestination-Exception *(p. 1190)*	- if an invalid destination is specified.

Implemented in **activemq::core::ActiveMQSession** (p. 264), and **activemq::cmsutil::PooledSession** (p. 1625).

Referenced by activemq::cmsutil::PooledSession::createConsumer().

6.469.4.8 virtual MessageConsumer* cms::Session::createConsumer (const Destination * *destination,* const std::string & *selector*) [pure virtual]

Creates a **MessageConsumer** (p. 1455) for the specified destination, using a message selector.

Parameters

destination	the **Destination** (p. 936) that this consumer receiving messages for.
selector	the **Message** (p. 1426) Selector to use

Returns

pointer to a new **MessageConsumer** (p. 1455) that is owned by the caller (caller deletes)

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
InvalidDestination-Exception *(p. 1190)*	- if an invalid destination is specified.
InvalidSelectorException *(p. 1195)*	- if the message selector is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 264), and **activemq::cmsutil::PooledSession** (p. 1626).

6.469.4.9 virtual MessageConsumer* cms::Session::createConsumer (const Destination * *destination,* const std::string & *selector,* bool *noLocal*) [pure virtual]

Creates a **MessageConsumer** (p. 1455) for the specified destination, using a message selector.

Parameters

destination	the **Destination** (p. 936) that this consumer receiving messages for.
selector	the **Message** (p. 1426) 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. 1455) that is owned by the caller (caller deletes)

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
InvalidDestination-Exception *(p. 1190)*	- if an invalid destination is specified.
InvalidSelectorException *(p. 1195)*	- if the message selector is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 265), and **activemq::cmsutil::PooledSession** (p. 1626).

6.469.4.10 virtual MessageConsumer* cms::Session::createDurableConsumer (const Topic * *destination,* const std::string & *name,* const std::string & *selector,* bool *noLocal* = false) [pure virtual]

Creates a durable subscriber to the specified topic, using a **Message** (p. 1426) 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. 1426) 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. 1455) that is owned by the caller (caller deletes)

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
InvalidDestination-Exception *(p. 1190)*	- if an invalid destination is specified.
InvalidSelectorException *(p. 1195)*	- if the message selector is invalid.

Implemented in **activemq::core::ActiveMQSession** (p. 265), and **activemq::cmsutil::PooledSession** (p. 1627).

Referenced by activemq::cmsutil::PooledSession::createDurableConsumer().

6.469.4.11 virtual MapMessage* cms::Session::createMapMessage ( ) [pure virtual]

Creates a new **MapMessage** (p. 1379).

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 266), and **activemq::cmsutil::PooledSession** (p. 1627).

Referenced by activemq::cmsutil::PooledSession::createMapMessage().

6.469.4.12 virtual Message* cms::Session::createMessage ( ) [pure virtual]

Creates a new **Message** (p. 1426).

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 266), and **activemq::cmsutil::PooledSession** (p. 1627).

Referenced by activemq::cmsutil::PooledSession::createMessage().

6.469.4.13 virtual MessageProducer* cms::Session::createProducer (const Destination * *destination* = NULL) [pure virtual]

Creates a **MessageProducer** (p. 1491) to send messages to the specified destination.

Parameters

destination	the **Destination** (p. 936) to send on

Returns

New **MessageProducer** (p. 1491) that is owned by the caller.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
InvalidDestination-Exception *(p. 1190)*	- if an invalid destination is specified.

Implemented in **activemq::core::ActiveMQSession** (p. 266), and **activemq::cmsutil::PooledSession** (p. 1628).

Referenced by activemq::cmsutil::PooledSession::createProducer().

6.469.4.14 virtual Queue* cms::Session::createQueue (const std::string & *queueName*) [pure virtual]

Creates a queue identity given a **Queue** (p. 1722) name.

Parameters

queueName	the name of the new **Queue** (p. 1722)

Returns

new **Queue** (p. 1722) pointer that is owned by the caller.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 266), and **activemq::cmsutil::PooledSession** (p. 1628).

Referenced by activemq::cmsutil::PooledSession::createQueue().

6.469.4.15 virtual StreamMessage* cms::Session::createStreamMessage ( ) [pure virtual]

Creates a new **StreamMessage** (p. 2020).

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 267), and **activemq::cmsutil::PooledSession** (p. 1628).

Referenced by activemq::cmsutil::PooledSession::createStreamMessage().

6.469.4.16 virtual TemporaryQueue* cms::Session::createTemporaryQueue ( ) [pure virtual]

Creates a **TemporaryQueue** (p. 2090) object.

Returns

new **TemporaryQueue** (p. 2090) pointer that is owned by the caller.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 267), and **activemq::cmsutil::PooledSession** (p. 1629).

Referenced by activemq::cmsutil::PooledSession::createTemporaryQueue().

6.469.4.17 virtual TemporaryTopic* cms::Session::createTemporaryTopic ( ) [pure virtual]

Creates a **TemporaryTopic** (p. 2091) object.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 267), and **activemq::cmsutil::PooledSession** (p. 1629).

Referenced by activemq::cmsutil::PooledSession::createTemporaryTopic().

6.469.4.18 virtual TextMessage* cms::Session::createTextMessage ( ) [pure virtual]

Creates a new **TextMessage** (p. 2093).

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 267), and **activemq::cmsutil::PooledSession** (p. 1629).

Referenced by activemq::cmsutil::PooledSession::createTextMessage().

6.469.4.19 virtual TextMessage* cms::Session::createTextMessage (const std::string & *text*) [pure virtual]

Creates a new **TextMessage** (p. 2093) and set the text to the value given.

Parameters

text	the initial text for the message

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 268), and **activemq::cmsutil::PooledSession** (p. 1629).

6.469.4.20 virtual Topic* cms::Session::createTopic (const std::string & *topicName*) [pure virtual]

Creates a topic identity given a **Queue** (p. 1722) name.

Parameters

topicName	the name of the new **Topic** (p. 2141)

Returns

new **Topic** (p. 2141) pointer that is owned by the caller.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 268), and **activemq::cmsutil::PooledSession** (p. 1630).

Referenced by activemq::cmsutil::PooledSession::createTopic().

6.469.4.21 virtual AcknowledgeMode cms::Session::getAcknowledgeMode ( ) const [pure virtual]

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 269), and **activemq::cmsutil::PooledSession** (p. 1630).

Referenced by activemq::cmsutil::PooledSession::getAcknowledgeMode().

6.469.4.22 virtual bool cms::Session::isTransacted ( ) const [pure virtual]

Gets if the Sessions is a Transacted **Session** (p. 1830).

Returns

transacted true - false.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 271), **activemq::cmsutil::PooledSession** (p. 1631), and **activemq::core::ActiveMQXASession** (p. 340).

Referenced by activemq::cmsutil::PooledSession::isTransacted().

6.469.4.23 virtual void cms::Session::recover ( ) [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

CMSException *(p. 640)*	- if the CMS provider fails to stop and restart message delivery due to some internal error.
IllegalStateException *(p. 1098)*	- if the method is called by a transacted session.

Implemented in **activemq::core::ActiveMQSession** (p. 272), and **activemq::cmsutil::PooledSession** (p. 1631).

Referenced by activemq::cmsutil::PooledSession::recover().

6.469.4.24 virtual void cms::Session::rollback ( ) [pure virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.
IllegalStateException *(p. 1098)*	- if the method is not called by a transacted session.

Implemented in **activemq::core::ActiveMQSession** (p. 273), **activemq::cmsutil::PooledSession** (p. 1631), and **activemq::core::ActiveMQXASession** (p. 340).

Referenced by activemq::cmsutil::PooledSession::rollback().

6.469.4.25 virtual void cms::Session::unsubscribe (const std::string & *name*) [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. 1455) 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 *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::core::ActiveMQSession** (p. 274), and **activemq::cmsutil::PooledSession** (p. 1632).

Referenced by activemq::cmsutil::PooledSession::unsubscribe().

The documentation for this class was generated from the following file:

- src/main/cms/**Session.h**

6.470 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

  Execute any number of operations against the supplied CMS session.

6.470.1 Detailed Description

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

6.470.2 Constructor & Destructor Documentation

6.470.2.1 virtual activemq::cmsutil::SessionCallback::~SessionCallback ( ) [inline, virtual]

6.470.3 Member Function Documentation

6.470.3.1 virtual void activemq::cmsutil::SessionCallback::doInCms (cms::Session * *session*) [pure virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session	the CMS `Session`

Exceptions

CMSException	if thrown by CMS API methods

Implemented in **activemq::cmsutil::CmsTemplate::ReceiveExecutor** (p. 1743), and **activemq::cmsutil::CmsTemplate::ProducerExecutor** (p. 1690).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**SessionCallback.h**

6.471 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **SessionId** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗**value**) **const**

  *Compares the **DataStructure** (p. 877) 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**

6.471.1 Member Typedef Documentation

6.471.1.1 typedef decaf::lang::PointerComparator<SessionId> activemq::commands::SessionId::COMPARATOR

6.471.2 Constructor & Destructor Documentation

6.471.2.1 activemq::commands::SessionId::SessionId ( )

6.471.2.2 activemq::commands::SessionId::SessionId (const SessionId & *other*)

6.471.2.3 activemq::commands::SessionId::SessionId (const ConnectionId * *connectionId,* long long *sessionId*)

6.471.2.4 activemq::commands::SessionId::SessionId (const ProducerId * *producerId*)

6.471.2.5 activemq::commands::SessionId::SessionId (const ConsumerId * *consumerId*)

6.471.2.6 virtual activemq::commands::SessionId::~SessionId ( ) [virtual]

6.471.3 Member Function Documentation

6.471.3.1 virtual SessionId* activemq::commands::SessionId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.471.3.2 virtual int activemq::commands::SessionId::compareTo (const SessionId & *value*) const [virtual]

6.471.3.3 virtual void activemq::commands::SessionId::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. 879).

6.471.3.4 virtual bool activemq::commands::SessionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.471.3.5 virtual bool activemq::commands::SessionId::equals (const SessionId & *value*) const [virtual]

6.471.3.6 virtual const std::string& activemq::commands::SessionId::getConnectionId ( ) const [virtual]

6.471.3.7 virtual std::string& activemq::commands::SessionId::getConnectionId ( ) [virtual]

6.471.3.8 virtual unsigned char activemq::commands::SessionId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.471.3.9 const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId ( ) const

6.471.3.10 virtual long long activemq::commands::SessionId::getValue ( ) const [virtual]

6.471.3.11 virtual bool activemq::commands::SessionId::operator< (const SessionId & *value*) const [virtual]

6.471.3.12 SessionId& activemq::commands::SessionId::operator= (const SessionId & *other*)

6.471.3.13 virtual bool activemq::commands::SessionId::operator== (const SessionId & *value*) const [virtual]

6.471.3.14 virtual void activemq::commands::SessionId::setConnectionId (const std::string & *connectionId*) [virtual]

6.471.3.15 virtual void activemq::commands::SessionId::setValue (long long *value*) [virtual]

6.471.3.16 virtual std::string activemq::commands::SessionId::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.471.4 Field Documentation

6.471.4.1 std::string activemq::commands::SessionId::connectionId [protected]

6.471.4.2 const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.471.4.3 long long activemq::commands::SessionId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**SessionId.h**

6.472 activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller Class Reference

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

`#include <src/main/activemq/wireformat/openwire/marshal/generated/SessionIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual ∼**SessionIdMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.472.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (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

6.472.2 Constructor & Destructor Documentation

6.472.2.1 activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::SessionIdMarshaller ( ) [inline]

6.472.2.2 virtual activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::∼SessionIdMarshaller ( ) [inline, virtual]

6.472.3 Member Function Documentation

6.472.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.472.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.472.3.3 virtual void activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.472.3.4 virtual void activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.472.3.5 virtual int activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.472.3.6 virtual void activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.472.3.7 virtual void activemq::wireformat::openwire::marshal::generated::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**SessionIdMarshaller.h**

6.473 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **SessionInfo** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- unsigned int **getAckMode** () **const**
- void **setAckMode** (unsigned int mode)
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () **const**
- 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)

  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 Attributes

- **Pointer**< **SessionId** > **sessionId**

6.473.1 Constructor & Destructor Documentation

6.473.1.1 activemq::commands::SessionInfo::SessionInfo ( )

6.473.1.2 virtual activemq::commands::SessionInfo::~SessionInfo ( ) [virtual]

6.473.2 Member Function Documentation

6.473.2.1 virtual SessionInfo* activemq::commands::SessionInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.473.2.2 virtual void activemq::commands::SessionInfo::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. 382).

6.473.2.3 Pointer<RemoveInfo> activemq::commands::SessionInfo::createRemoveCommand ( ) const

6.473.2.4 virtual bool activemq::commands::SessionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.473.2.5 unsigned int activemq::commands::SessionInfo::getAckMode ( ) const [inline]

6.473.2.6 virtual unsigned char activemq::commands::SessionInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.473.2.7 virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ( ) const [virtual]

Referenced by activemq::state::ConnectionState::addSession().

6.473.2.8 virtual Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ( ) [virtual]

6.473.2.9 void activemq::commands::SessionInfo::setAckMode (unsigned int *mode*) [inline]

6.473.2.10 virtual void activemq::commands::SessionInfo::setSessionId (const Pointer< SessionId > & *sessionId*) [virtual]

6.473.2.11 virtual std::string activemq::commands::SessionInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.473.2.12 virtual Pointer<Command> activemq::commands::SessionInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.473.3 Field Documentation

6.473.3.1 const unsigned char activemq::commands::SessionInfo::ID_SESSIONINFO = 4 [static]

6.473.3.2 Pointer<SessionId> activemq::commands::SessionInfo::sessionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**SessionInfo.h**

6.474 activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 1851).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/SessionInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual ∼**SessionInfoMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.474.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 1851).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.474.2 Constructor & Destructor Documentation

6.474.2.1 activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]

6.474.2.2 virtual activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::∼SessionInfoMarshaller ( ) [inline, virtual]

6.474.3 Member Function Documentation

6.474.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.474.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.474.3.3 virtual void activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.474.3.4 virtual void activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.474.3.5 virtual int activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.474.3.6 virtual void activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.474.3.7 virtual void activemq::wireformat::openwire::marshal::generated::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**SessionInfoMarshaller.h**

6.475 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** ()

  Takes a session from the pool, creating one if necessary.
- virtual void **returnSession** (**PooledSession** *session)

  Returns a session to the pool.
- **ResourceLifecycleManager** * **getResourceLifecycleManager** ()

6.475.1 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. 1778), not by this pool. This class is thread-safe.

6.475.2 Constructor & Destructor Documentation

6.475.2.1 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
resource-Lifecycle-Manager	the object responsible for managing the lifecycle of any allocated **cms::Session** (p. 1830) resources.

6.475.2.2 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. 1778).

6.475.3 Member Function Documentation

6.475.3.1 ResourceLifecycleManager* activemq::cmsutil::SessionPool::getResourceLifecycleManager ( ) [inline]

6.475.3.2 virtual void activemq::cmsutil::SessionPool::returnSession (PooledSession * *session*) [virtual]

Returns a session to the pool.

Parameters

session	the session to be returned.

6.475.3.3 virtual PooledSession* activemq::cmsutil::SessionPool::takeSession ( ) [virtual]

Takes a session from the pool, creating one if necessary.

Returns

the pooled session object

Exceptions

CMSException	if an error occurred

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**SessionPool.h**

6.476 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** ()

6.476.1 Constructor & Destructor Documentation

6.476.1.1 activemq::state::SessionState::SessionState (const Pointer< SessionInfo > & *info*)

6.476.1.2 virtual activemq::state::SessionState::~SessionState ( ) [virtual]

6.476.2 Member Function Documentation

6.476.2.1 void activemq::state::SessionState::addConsumer (const Pointer< ConsumerInfo > & *info*) [inline]

References activemq::commands::ConsumerInfo::getConsumerId().

6.476.2.2 void activemq::state::SessionState::addProducer (const Pointer< ProducerInfo > & *info*)

6.476.2.3 void activemq::state::SessionState::checkShutdown ( ) const

6.476.2.4 Pointer<ConsumerState> activemq::state::SessionState::getConsumerState (const Pointer< ConsumerId > & *id*) [inline]

6.476.2.5 std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates ( ) const [inline]

6.476.2.6 const Pointer<SessionInfo> activemq::state::SessionState::getInfo ( ) const [inline]

6.476.2.7 Pointer<ProducerState> activemq::state::SessionState::getProducerState (const Pointer< ProducerId > & *id*) [inline]

6.476.2.8 std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates ( ) const [inline]

6.476.2.9 Pointer<ConsumerState> activemq::state::SessionState::removeConsumer (const Pointer< ConsumerId > & *id*) [inline]

6.476.2.10 Pointer<ProducerState> activemq::state::SessionState::removeProducer (const Pointer< ProducerId > & *id*)

6.476.2.11 void activemq::state::SessionState::shutdown ( ) [inline]

6.476.2.12 std::string activemq::state::SessionState::toString ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**SessionState.h**

6.477 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** ()

6.477.1 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

6.477.2 Constructor & Destructor Documentation

6.477.2.1 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**

6.478 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)
- virtual ~**Short** ()
- virtual int **compareTo** (**const Short** &s) **const**

  *Compares this **Short*** (p. *1858*) *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. *1858*) *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)

  *Decodes a **String** (p. 2031) into a **Short** (p. 1858).*
- 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)

  Parses the string argument as a signed short in the radix specified by the second argument.
- static short **parseShort** (**const** std::string &s)

  Parses the string argument as a signed decimal short.
- static **Short valueOf** (short value)

  *Returns a **Short** (p. 1858) instance representing the specified short value.*
- static **Short valueOf** (**const** std::string &value)

  *Returns a **Short** (p. 1858) object holding the value given by the specified std::string.*
- static **Short valueOf** (**const** std::string &value, int radix)

  *Returns a **Short** (p. 1858) 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.

6.478.1 Constructor & Destructor Documentation

6.478.1.1 decaf::lang::Short::Short (short *value*)

Parameters

value	- short to wrap

6.478.1.2 decaf::lang::Short::Short (const std::string & *value*)

Parameters

value	The string value to convert to short and wrap.

Exceptions

NumberFormatException	if the string is not well formed number value.

6.478.1.3 virtual decaf::lang::Short::~Short ( ) [inline, virtual]

6.478.2 Member Function Documentation

6.478.2.1 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.

Reimplemented from **decaf::lang::Number** (p. 1544).

6.478.2.2 virtual int decaf::lang::Short::compareTo (const Short & *s*) const [virtual]

Compares this **Short** (p. 1858) instance with another.

Parameters

s	- the **Short** (p. 1858) 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. 687).

6.478.2.3 virtual int decaf::lang::Short::compareTo (const short & *s*) const [virtual]

Compares this **Short** (p. 1858) instance with another.

Parameters

s	- the **Short** (p. 1858) 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. 687).

6.478.2.4 static Short decaf::lang::Short::decode (const std::string & *value*) [static]

Decodes a **String** (p. 2031) into a **Short** (p. 1858).

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. 1863) 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** (p. 2031) is the minus sign. No whitespace characters are permitted in the string.

Parameters

value	- The string to decode

Returns

a **Short** (p. 1858) object containing the decoded value

Exceptions

NumberFomatException	if the string is not formatted correctly.

6.478.2.5 virtual double decaf::lang::Short::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.478.2.6 bool decaf::lang::Short::equals (const Short & *s*) const [inline, virtual]

Returns

true if the two **Short** (p. 1858) Objects have the same value.

Implements **decaf::lang::Comparable**< **Short** > (p. 688).

6.478.2.7 bool decaf::lang::Short::equals (const short & *s*) const [inline, virtual]

Returns

true if the two **Short** (p. 1858) Objects have the same value.

Implements **decaf::lang::Comparable**< **short** > (p. 688).

6.478.2.8 virtual float decaf::lang::Short::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.478.2.9 virtual int decaf::lang::Short::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1544).

6.478.2.10 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.

Implements **decaf::lang::Number** (p. 1545).

6.478.2.11 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.

Implements **decaf::lang::Comparable< Short >** (p. 688).

6.478.2.12 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. 688).

6.478.2.13 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.

Implements **decaf::lang::Comparable**< **Short** > (p. 688).

6.478.2.14 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. 688).

6.478.2.15 static short decaf::lang::Short::parseShort (const std::string & *s*, int *radix*) [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. 595) 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. 599) or larger than **Character.MAX_RADIX** (p. 599).
- 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** (p. 2031) 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** (p. 2031) does not contain a parsable short.

6.478.2.16 static short decaf::lang::Short::parseShort (const std::string & *s*) [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** (p. 2031) to convert to a short

Returns

the converted short value

Exceptions

NumberFormatException	if the string is not a short.

6.478.2.17 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.

6.478.2.18 virtual short decaf::lang::Short::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1545).

6.478.2.19 std::string decaf::lang::Short::toString ( ) const

Returns

this **Short** (p. 1858) Object as a **String** (p. 2031) Representation

6.478.2.20 static std::string decaf::lang::Short::toString (short *value*) [static]

Returns

a string representing the primitive value as Base 10

6.478.2.21 static Short decaf::lang::Short::valueOf (short *value*) [static]

Returns a **Short** (p. 1858) instance representing the specified short value.

Parameters

value	- the short to wrap

Returns

the new **Short** (p. 1858) object wrapping value.

6.478.2.22 static Short decaf::lang::Short::valueOf (const std::string & *value*) [static]

Returns a **Short** (p. 1858) 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. 1858) object that represents the short value specified by the string.

Parameters

value	- std::string to parse as base 10

Returns

new **Short** (p. 1858) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a decimal short.

6.478.2.23 static Short decaf::lang::Short::valueOf (const std::string & *value,* int *radix*) [static]

Returns a **Short** (p. 1858) 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. 1858) 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. 1858) Object wrapping the primitive

Exceptions

NumberFormatException	if the string is not a valid short.

6.478.3 Field Documentation

6.478.3.1 const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF [static]

Max Value for this Object's primitive type.

6.478.3.2 const short decaf::lang::Short::MIN_VALUE = (short)0x8000 [static]

Max Value for this Object's primitive type.

6.478.3.3 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**

6.479 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** (int size, bool readOnly=false)

  *Creates a **ShortArrayBuffer** (p. 1866) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ShortArrayBuffer** (short ∗**array**, int size, int offset, int length, bool readOnly=false)

  *Creates a **ShortArrayBuffer** (p. 1866) object that wraps the given array.*
- **ShortArrayBuffer** (**const decaf::lang::Pointer**< **ByteArrayAdapter** > &**array**, int offset, int length, bool readOnly=false)

  Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.
- **ShortArrayBuffer** (**const ShortArrayBuffer** &other)

  *Create a **ShortArrayBuffer** (p. 1866) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ∼**ShortArrayBuffer** ()
- virtual short ∗ **array** ()

  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. 451)*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual int **arrayOffset** ()

  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. 1739)	*if this **Buffer** (p. 451) is read only.*
UnsupportedOperation-Exception	*if the underlying store has no array.*

- virtual **ShortBuffer** * **asReadOnlyBuffer** () **const**

  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.

- virtual **ShortBuffer** & **compact** ()

  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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1874).*

  Exceptions

ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ShortBuffer** * **duplicate** ()

  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. 451) which the caller owns.*

- virtual short **get** ()

  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. 474)	*if there no more data to return.*

- virtual short **get** (int index) **const**

  Absolute get method.
  Reads the value at the given index.

Parameters

index	*The index in the **Buffer** (p. 451) 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, or the index is negative.*

- virtual bool **hasArray** () **const**

  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. Subclasses 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.

- virtual bool **isReadOnly** () **const**

  Tells whether or not this buffer is read-only.

  Returns

  true if, and only if, this buffer is read-only.

- virtual **ShortBuffer** & **put** (short value)

  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. 472)	*if this buffer's current position is not smaller than its limit.*
ReadOnlyBufferException (p. 1739)	*if this buffer is read-only.*

- virtual **ShortBuffer** & **put** (int index, short value)

  Writes the given shorts into this buffer at the given index.

  Parameters

index	*The position in the **Buffer** (p. 451) 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. 1739)	*if this buffer is read-only.*

- virtual **ShortBuffer** * **slice** () **const**

  *Creates a new **ShortBuffer** (p. 1874) 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. *1874) which the caller owns.*

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  Sets this ***ShortArrayBuffer*** *(*p. *1866) as Read-Only.*

6.479.1 Constructor & Destructor Documentation

6.479.1.1 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (int *size*, bool *readOnly* = `false`)

Creates a **ShortArrayBuffer** (p. 1866) 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

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

IllegalArguementException	if the capacity value is negative.

6.479.1.2 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (short * *array*, int *size*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a **ShortArrayBuffer** (p. 1866) 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	The actual array to wrap.
size	The size of the given array.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if buffer is NULL
IndexOutOfBounds-Exception	if offset is greater than array capacity.

6.479.1.3 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array*, int *offset*, int *length*, bool *readOnly* = `false`)

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **ShortArrayBuffer** (p. 1866) will be that of the remaining capacity of the passed buffer.

Parameters

array	The ByteArrayAdapter to wrap.
offset	The position that is this buffers start position.
length	The limit of how many bytes into the array this Buffer can write.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

Exceptions

NullPointerException	if array is NULL
IndexOutOfBounds-Exception	if offset + length is greater than array size.

6.479.1.4 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const ShortArrayBuffer & *other*)

Create a **ShortArrayBuffer** (p. 1866) that mirrors this one, meaning it shares a reference to this buffers ByteArray-Adapter and when changes are made to that data it is reflected in both.

Parameters

other	The **ShortArrayBuffer** (p. 1866) this one is to mirror.

6.479.1.5 virtual decaf::internal::nio::ShortArrayBuffer::~ShortArrayBuffer ( ) [virtual]

6.479.2 Member Function Documentation

6.479.2.1 virtual short* decaf::internal::nio::ShortArrayBuffer::array ( ) [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. 451)

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 1876).

6.479.2.2 virtual int decaf::internal::nio::ShortArrayBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 1877).

6.479.2.3 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. 1877).

6.479.2.4 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1874).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 1877).

6.479.2.5 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** (p. 451) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 1878).

6.479.2.6 virtual short decaf::internal::nio::ShortArrayBuffer::get ( ) [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. 474)*	if there no more data to return.

Implements **decaf::nio::ShortBuffer** (p. 1878).

6.479.2.7 virtual short decaf::internal::nio::ShortArrayBuffer::get (int *index*) const [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the short is to be read.

Returns

the short that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or the index is negative.

Implements **decaf::nio::ShortBuffer** (p. 1878).

6.479.2.8 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. Subclasses 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. 1880).

6.479.2.9 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.

Implements **decaf::nio::Buffer** (p. 454).

6.479.2.10 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (short *value*) [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. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 1881).

6.479.2.11 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (int *index*, short *value*) [virtual]

Writes the given shorts into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The shorts to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 1882).

6.479.2.12 virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ShortArrayBuffer** (p. 1866) as Read-Only.

Parameters

value	Boolean value, true if this buffer is to be read-only, false otherwise.

6.479.2.13 virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::slice ( ) const [virtual]

Creates a new **ShortBuffer** (p. 1874) 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. 1874) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 1882).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ShortArrayBuffer.h**

6.480 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

  Returns the short array that backs this buffer (optional operation).
- virtual int **arrayOffset** ()=0

  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

  Compacts this buffer.
- virtual **ShortBuffer** * **duplicate** ()=0

  Creates a new short buffer that shares this buffer's content.

- virtual short **get** ()=0

  Relative get method.
- virtual short **get** (int index) **const** =0

  Absolute get method.
- **ShortBuffer** & **get** (std::vector< short > buffer)

  Relative bulk get method.
- **ShortBuffer** & **get** (short *buffer, int size, int offset, int length)

  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)

  This method transfers the shorts remaining in the given source buffer into this buffer.
- **ShortBuffer** & **put** (**const** short *buffer, int size, int offset, int length)

  This method transfers shorts into this buffer from the given source array.
- **ShortBuffer** & **put** (std::vector< short > &buffer)

  This method transfers the entire content of the given source shorts array into this buffer.
- virtual **ShortBuffer** & **put** (short value)=0

  Writes the given shorts into this buffer at the current position, and then increments the position.
- virtual **ShortBuffer** & **put** (int index, short value)=0

  Writes the given shorts into this buffer at the given index.
- virtual **ShortBuffer** * **slice** () **const** =0

  *Creates a new **ShortBuffer** (p. 1874) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (**const ShortBuffer** &value) **const**
- virtual bool **equals** (**const ShortBuffer** &value) **const**
- virtual bool **operator==** (**const ShortBuffer** &value) **const**
- virtual bool **operator<** (**const ShortBuffer** &value) **const**

Static Public Member Functions

- static **ShortBuffer** * **allocate** (int **capacity**)

  Allocates a new Double buffer.
- static **ShortBuffer** * **wrap** (short ***array**, int size, int offset, int length)

  *Wraps the passed buffer with a new **ShortBuffer** (p. 1874).*
- static **ShortBuffer** * **wrap** (std::vector< short > &buffer)

  *Wraps the passed STL short Vector in a **ShortBuffer** (p. 1874).*

Protected Member Functions

- **ShortBuffer** (int **capacity**)

  *Creates a **ShortBuffer** (p. 1874) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.480.1 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.

6.480.2 Constructor & Destructor Documentation

6.480.2.1 decaf::nio::ShortBuffer::ShortBuffer (int *capacity*) [protected]

Creates a **ShortBuffer** (p. 1874) 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	The size and limit of the **Buffer** (p. 451) in doubles

Exceptions

IllegalArguementException	if capacity is negative.

6.480.2.2 virtual decaf::nio::ShortBuffer::~ShortBuffer ( ) [inline, virtual]

6.480.3 Member Function Documentation

6.480.3.1 static ShortBuffer* decaf::nio::ShortBuffer::allocate (int *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. 1874) that was allocated, caller owns.

6.480.3.2 virtual short* decaf::nio::ShortBuffer::array ( ) [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. 451)

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1870).

6.480.3.3 virtual int decaf::nio::ShortBuffer::arrayOffset ( ) [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. 1739)*	if this **Buffer** (p. 451) is read only.
UnsupportedOperation-Exception	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1870).

6.480.3.4 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. 1871).

6.480.3.5 virtual ShortBuffer& decaf::nio::ShortBuffer::compact ( ) [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. 455) 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. 455) - 1 is copied to index n = **limit()** (p. 455) - 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. 1874).

Exceptions

ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1871).

6.480.3.6 virtual int decaf::nio::ShortBuffer::compareTo (const ShortBuffer & *value*) const [virtual]

6.480.3.7 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. 451) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1871).

6.480.3.8 virtual bool decaf::nio::ShortBuffer::equals (const ShortBuffer & *value*) const [virtual]

6.480.3.9 virtual short decaf::nio::ShortBuffer::get ( ) [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. 474)*	if there no more data to return.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1872).

6.480.3.10 virtual short decaf::nio::ShortBuffer::get (int *index*) const [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index	The index in the **Buffer** (p. 451) where the short is to be read.

Returns

the short that is located at the given index.

Exceptions

IndexOutOfBounds-Exception	if index is not smaller than the buffer's limit, or the index is negative.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1872).

6.480.3.11 ShortBuffer& decaf::nio::ShortBuffer::get (std::vector< short > *buffer*)

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. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length shorts remaining in this buffer.

6.480.3.12 ShortBuffer& decaf::nio::ShortBuffer::get (short * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no bytes are transferred and a **BufferUnderflowException** (p. 474) 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	The pointer to an allocated buffer to fill.
size	The size of the buffer provided.
offset	The position in the buffer to start filling.
length	The amount of data to put in the passed buffer.

Returns

a reference to this **Buffer** (p. 451).

Exceptions

BufferUnderflowException *(p. 474)*	if there are fewer than length shorts remaining in this buffer
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.480.3.13 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. Subclasses 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. 1872).

6.480.3.14 virtual bool decaf::nio::ShortBuffer::operator< (const ShortBuffer & *value*) const [virtual]

6.480.3.15 virtual bool decaf::nio::ShortBuffer::operator== (const ShortBuffer & *value*) const [virtual]

6.480.3.16 ShortBuffer& decaf::nio::ShortBuffer::put (ShortBuffer & *src*)

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. 456), then no shorts are transferred and a **BufferOverflowException** (p. 472) 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. 472)*	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. 1739)*	if this buffer is read-only.

6.480.3.17 ShortBuffer& decaf::nio::ShortBuffer::put (const short * *buffer,* int *size,* int *offset,* int *length*)

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. 456), then no shorts are transferred and a **BufferOverflowException** (p. 472) 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.
size	The size of the buffer passed.
offset	The offset within the array of the first char to be read.
length	The number of shorts to be read from the given array.

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only
NullPointerException	if the passed buffer is null.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.480.3.18 ShortBuffer& decaf::nio::ShortBuffer::put (std::vector< short > & *buffer*)

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. 1874).

Returns

a reference to this buffer.

Exceptions

BufferOverflowException *(p. 472)*	if there is insufficient space in this buffer.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

6.480.3.19 virtual ShortBuffer& decaf::nio::ShortBuffer::put (short *value*) [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. 472)*	if this buffer's current position is not smaller than its limit.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1873).

6.480.3.20 virtual ShortBuffer& decaf::nio::ShortBuffer::put (int *index*, short *value*) [pure virtual]

Writes the given shorts into this buffer at the given index.

Parameters

index	The position in the **Buffer** (p. 451) to write the data.
value	The shorts to write.

Returns

a reference to this buffer.

Exceptions

IndexOutOfBounds-Exception	if index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException *(p. 1739)*	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1873).

6.480.3.21 virtual ShortBuffer* decaf::nio::ShortBuffer::slice ( ) const [pure virtual]

Creates a new **ShortBuffer** (p. 1874) 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. 1874) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 1874).

6.480.3.22 virtual std::string decaf::nio::ShortBuffer::toString ( ) const [virtual]

Returns

a std::string describing this object

6.480.3.23 static ShortBuffer* decaf::nio::ShortBuffer::wrap (short * *array,* int *size,* int *offset,* int *length*) [static]

Wraps the passed buffer with a new **ShortBuffer** (p. 1874).

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.
size	The size of the passed in array.
offset	The offset of the subarray to be used.
length	The length of the subarray to be used.

Returns

a new **ShortBuffer** (p. 1874) that is backed by buffer, caller owns.

Exceptions

NullPointerException	if the array pointer is NULL.
IndexOutOfBounds-Exception	if the preconditions of size, offset, or length are not met.

6.480.3.24 static ShortBuffer* decaf::nio::ShortBuffer::wrap (std::vector< short > & *buffer*) [static]

Wraps the passed STL short Vector in a **ShortBuffer** (p. 1874).

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. 1874) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ShortBuffer.h**

6.481 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **ShutdownInfo** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual bool **isShutdownInfo** () **const**
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ∗visitor)

  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

6.481.1 Constructor & Destructor Documentation

6.481.1.1 activemq::commands::ShutdownInfo::ShutdownInfo ( )

6.481.1.2 virtual activemq::commands::ShutdownInfo::∼ShutdownInfo ( ) [virtual]

6.481.2 Member Function Documentation

6.481.2.1 virtual ShutdownInfo∗ activemq::commands::ShutdownInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.481.2.2 virtual void activemq::commands::ShutdownInfo::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. 382).

6.481.2.3 virtual bool activemq::commands::ShutdownInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.481.2.4 virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.481.2.5 virtual bool activemq::commands::ShutdownInfo::isShutdownInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isShutdownInfo()** (p. 1885) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 385).

6.481.2.6 virtual std::string activemq::commands::ShutdownInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.481.2.7 virtual Pointer<Command> activemq::commands::ShutdownInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.481.3 Field Documentation

6.481.3.1 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**

6.482 activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 1886).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/ShutdownInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual ~**ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.482.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (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

6.482.2 Constructor & Destructor Documentation

6.482.2.1 activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::ShutdownInfoMarshaller ( ) [inline]

6.482.2.2 virtual activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::~ShutdownInfoMarshaller ( ) [inline, virtual]

6.482.3 Member Function Documentation

6.482.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.482.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.482.3.3 virtual void activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.482.3.4 virtual void activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.482.3.5 virtual int activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.482.3.6 virtual void activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.482.3.7 virtual void activemq::wireformat::openwire::marshal::generated::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**ShutdownInfoMarshaller.h**

6.483 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 ()

6.483.1 Constructor & Destructor Documentation

6.483.1.1 decaf::security::SignatureException::SignatureException ( ) throw () [inline]

Default Constructor.

6.483.1.2 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

6.483.1.3 decaf::security::SignatureException::SignatureException (const SignatureException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.483.1.4 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

6.483.1.5 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.

6.483.1.6 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

6.483.1.7 virtual decaf::security::SignatureException::~SignatureException ( ) throw () [inline, virtual]

6.483.2 Member Function Documentation

6.483.2.1 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. 1081).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SignatureException.h**

6.484 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 1333) 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.

6.484.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 1333) in a human readable format.

The summary will typically be 1 or 2 lines.

Since

1.0

6.484.2 Constructor & Destructor Documentation

6.484.2.1 decaf::util::logging::SimpleFormatter::SimpleFormatter ( )

6.484.2.2 virtual decaf::util::logging::SimpleFormatter::~SimpleFormatter ( ) [virtual]

6.484.3 Member Function Documentation

6.484.3.1 virtual std::string decaf::util::logging::SimpleFormatter::format (const LogRecord & *record*) const [virtual]

Format the given log record and return the formatted string.

Parameters

record	The Log Record to Format.

Implements **decaf::util::logging::Formatter** (p. 1074).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleFormatter.h**

6.485 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*** (p. *1253) Log.*
- virtual void **debug** (**const** std::string &file, **const** int line, **const** std::string &message)
  *Log a Debug **Level*** (p. *1253) Log.*
- virtual void **info** (**const** std::string &file, **const** int line, **const** std::string &message)
  *Log a Informational **Level*** (p. *1253) Log.*
- virtual void **warn** (**const** std::string &file, **const** int line, **const** std::string &message)
  *Log a Warning **Level*** (p. *1253) Log.*
- virtual void **error** (**const** std::string &file, **const** int line, **const** std::string &message)
  *Log a Error **Level*** (p. *1253) Log.*
- virtual void **fatal** (**const** std::string &file, **const** int line, **const** std::string &message)
  *Log a Fatal **Level*** (p. *1253) Log.*
- virtual void **log** (**const** std::string &message)
  No-frills log.

6.485.1 Constructor & Destructor Documentation

6.485.1.1 decaf::util::logging::SimpleLogger::SimpleLogger (const std::string & *name*)

Constructor.

6.485.1.2 virtual decaf::util::logging::SimpleLogger::~SimpleLogger ( ) [virtual]

Destructor.

6.485.2 Member Function Documentation

6.485.2.1 virtual void decaf::util::logging::SimpleLogger::debug (const std::string & *file,* const int *line,* const std::string & *message*) [virtual]

Log a Debug **Level** (p. 1253) Log.

6.485.2.2 virtual void decaf::util::logging::SimpleLogger::error (const std::string & *file,* const int *line,* const std::string & *message*) [virtual]

Log a Error **Level** (p. 1253) Log.

6.485.2.3 virtual void decaf::util::logging::SimpleLogger::fatal (const std::string & *file,* const int *line,* const std::string & *message*) [virtual]

Log a Fatal **Level** (p. 1253) Log.

6.485.2.4 virtual void decaf::util::logging::SimpleLogger::info (const std::string & *file,* const int *line,* const std::string & *message*) [virtual]

Log a Informational **Level** (p. 1253) Log.

6.485.2.5 virtual void decaf::util::logging::SimpleLogger::log (const std::string & *message*) [virtual]

No-frills log.

6.485.2.6 virtual void decaf::util::logging::SimpleLogger::mark (const std::string & *message*) [virtual]

Log a Mark Block **Level** (p. 1253) Log.

6.485.2.7 virtual void decaf::util::logging::SimpleLogger::warn (const std::string & *file,* const int *line,* const std::string & *message*) [virtual]

Log a Warning **Level** (p. 1253) Log.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleLogger.h**

6.486 activemq::core::SimplePriorityMessageDispatchChannel Class Reference

`#include <src/main/activemq/core/SimplePriorityMessageDispatchChannel.h>`

Inheritance diagram for activemq::core::SimplePriorityMessageDispatchChannel:

Public Member Functions

- **SimplePriorityMessageDispatchChannel** ()
- virtual ~**SimplePriorityMessageDispatchChannel** ()
- virtual void **enqueue** (**const Pointer**< **MessageDispatch** > &message)

  Add a Message to the Channel behind all pending message.
- virtual void **enqueueFirst** (**const Pointer**< **MessageDispatch** > &message)

  Add a message to the front of the Channel.
- virtual bool **isEmpty** () **const**
- virtual bool **isClosed** () **const**
- virtual bool **isRunning** () **const**
- virtual **Pointer**< **MessageDispatch** > **dequeue** (long long timeout)

  Used to get an enqueued message.
- virtual **Pointer**< **MessageDispatch** > **dequeueNoWait** ()

  Used to get an enqueued message if there is one queued right now.
- virtual **Pointer**< **MessageDispatch** > **peek** () **const**

  Peek in the Queue and return the first message in the Channel without removing it from the channel.
- virtual void **start** ()

  Starts dispatch of messages from the Channel.
- virtual void **stop** ()

  Stops dispatch of message from the Channel.
- virtual void **close** ()

  Close this channel no messages will be dispatched after this method is called.
- virtual void **clear** ()

  Clear the Channel, all pending messages are removed.
- virtual int **size** () **const**
- virtual 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.

6.486.1 Constructor & Destructor Documentation

6.486.1.1 activemq::core::SimplePriorityMessageDispatchChannel::SimplePriorityMessageDispatchChannel ( )

6.486.1.2 virtual activemq::core::SimplePriorityMessageDispatchChannel::~SimplePriorityMessageDispatchChannel ( ) [virtual]

6.486.2 Member Function Documentation

6.486.2.1 virtual void activemq::core::SimplePriorityMessageDispatchChannel::clear ( ) [virtual]

Clear the Channel, all pending messages are removed.

Implements **activemq::core::MessageDispatchChannel** (p. 1463).

6.486.2.2 virtual void activemq::core::SimplePriorityMessageDispatchChannel::close ( ) [virtual]

Close this channel no messages will be dispatched after this method is called.

Implements **activemq::core::MessageDispatchChannel** (p. 1463).

6.486.2.3 virtual Pointer<MessageDispatch> activemq::core::SimplePriorityMessageDispatchChannel::dequeue (long long *timeout*) [virtual]

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 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 consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

ActiveMQException	

Implements **activemq::core::MessageDispatchChannel** (p. 1463).

6.486.2.4 virtual Pointer<MessageDispatch> activemq::core::SimplePriorityMessageDispatchChannel::dequeueNoWait ( ) [virtual]

Used to get an enqueued message if there is one queued right now.

If there is no waiting message than this method returns Null.

Returns

a message if there is one in the queue.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.486.2.5 virtual void activemq::core::SimplePriorityMessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & *message*) [virtual]

Add a Message to the Channel behind all pending message.

Parameters

message	- The message to add to the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.486.2.6 virtual void activemq::core::SimplePriorityMessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & *message*) [virtual]

Add a message to the front of the Channel.

Parameters

message	- The Message to add to the front of the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.486.2.7 virtual bool activemq::core::SimplePriorityMessageDispatchChannel::isClosed ( ) const [inline, virtual]

Returns

has the Queue been closed.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.486.2.8 virtual bool activemq::core::SimplePriorityMessageDispatchChannel::isEmpty ( ) const [virtual]

Returns

true if there are no messages in the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1464).

6.486.2.9 virtual bool activemq::core::SimplePriorityMessageDispatchChannel::isRunning ( ) const [inline, virtual]

Returns

true if the Channel currently running and will dequeue message.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.486.2.10 virtual void activemq::core::SimplePriorityMessageDispatchChannel::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. 2046).

6.486.2.11 virtual void activemq::core::SimplePriorityMessageDispatchChannel::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. 2047).

6.486.2.12 virtual void activemq::core::SimplePriorityMessageDispatchChannel::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. 2048).

6.486.2.13 virtual Pointer<MessageDispatch> activemq::core::SimplePriorityMessageDispatchChannel::peek ( ) const [virtual]

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.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.486.2.14 virtual std::vector< Pointer<MessageDispatch> > activemq::core::SimplePriorityMessageDispatchChannel::removeAll ( ) [virtual]

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.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.486.2.15 virtual int activemq::core::SimplePriorityMessageDispatchChannel::size ( ) const [virtual]

Returns

the number of Messages currently in the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.486.2.16 virtual void activemq::core::SimplePriorityMessageDispatchChannel::start ( ) [virtual]

Starts dispatch of messages from the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.486.2.17 virtual void activemq::core::SimplePriorityMessageDispatchChannel::stop ( ) [virtual]

Stops dispatch of message from the Channel.

Implements **activemq::core::MessageDispatchChannel** (p. 1465).

6.486.2.18 virtual bool activemq::core::SimplePriorityMessageDispatchChannel::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. 2048).

6.486.2.19 virtual void activemq::core::SimplePriorityMessageDispatchChannel::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. 2049).

6.486.2.20 virtual void activemq::core::SimplePriorityMessageDispatchChannel::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.

IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.486.2.21 virtual void activemq::core::SimplePriorityMessageDispatchChannel::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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.486.2.22 virtual void activemq::core::SimplePriorityMessageDispatchChannel::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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**SimplePriorityMessageDispatchChannel.h**

6.487 decaf::net::Socket Class Reference

`#include <src/main/decaf/net/Socket.h>`

Inheritance diagram for decaf::net::Socket:

Public Member Functions

- **Socket** ()

  *Creates an unconnected **Socket** (p. 1900) using the set **SocketImplFactory** (p. 1928) or if non is set than the default SockImpl type is created.*
- **Socket** (**SocketImpl** ∗**impl**)

  *Creates a **Socket** (p. 1900) wrapping the provided **SocketImpl** (p. 1921) instance, this **Socket** (p. 1900) is considered unconnected.*
- **Socket** (**const InetAddress** ∗address, int port)

  *Creates a new **Socket** (p. 1900) instance and connects it to the given address and port.*
- **Socket** (**const InetAddress** ∗address, int port, **const InetAddress** ∗localAddress, int localPort)

  *Creates a new **Socket** (p. 1900) instance and connects it to the given address and port.*
- **Socket** (**const** std::string &host, int port)

  *Creates a new **Socket** (p. 1900) instance and connects it to the given host and port.*
- **Socket** (**const** std::string &host, int port, **const InetAddress** ∗localAddress, int localPort)

  *Creates a new **Socket** (p. 1900) instance and connects it to the given host and port.*
- virtual ∼**Socket** ()
- virtual void **bind** (**const** std::string &ipaddress, int port)

  *Binds this **Socket** (p. 1900) to the given local address and port.*
- virtual void **close** ()

  *Closes the **Socket** (p. 1900).*
- virtual void **connect** (**const** std::string &host, int port)

  Connects to the specified destination.
- virtual void **connect** (**const** std::string &host, int port, int timeout)

  Connects to the specified destination, with a specified timeout value.
- bool **isConnected** () **const**

  Indicates whether or not this socket is connected to am end point.
- bool **isClosed** () **const**
- bool **isBound** () **const**
- bool **isInputShutdown** () **const**
- bool **isOutputShutdown** () **const**
- virtual **decaf::io::InputStream** ∗ **getInputStream** ()

  Gets the InputStream for this socket if its connected.
- virtual **decaf::io::OutputStream** ∗ **getOutputStream** ()

  Gets the OutputStream for this socket if it is connected.
- int **getPort** () **const**

  *Gets the on the remote host this **Socket** (p. 1900) is connected to.*
- int **getLocalPort** () **const**

  Gets the local port the socket is bound to.
- std::string **getInetAddress** () **const**

  Returns the address to which the socket is connected.

- std::string **getLocalAddress** () **const**

  Gets the local address to which the socket is bound.
- virtual void **shutdownInput** ()

  Shuts down the InputStream for this socket essentially marking it as EOF.
- virtual void **shutdownOutput** ()

  Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OuputStream::write will throw an IOException.
- virtual int **getSoLinger** () **const**

  Gets the linger time for the socket, SO_LINGER.
- virtual void **setSoLinger** (bool state, int timeout)

  Sets the linger time (SO_LINGER) using a specified time value, this limits of this value are platform specific.
- virtual bool **getKeepAlive** () **const**

  Gets the keep alive flag for this socket, SO_KEEPALIVE.
- virtual void **setKeepAlive** (bool keepAlive)

  Enables/disables the keep alive flag for this socket, SO_KEEPALIVE.
- virtual int **getReceiveBufferSize** () **const**

  Gets the receive buffer size for this socket, SO_RCVBUF.
- virtual void **setReceiveBufferSize** (int size)

  Sets the receive buffer size for this socket, SO_RCVBUF.
- virtual bool **getReuseAddress** () **const**

  Gets the reuse address flag, SO_REUSEADDR.
- virtual void **setReuseAddress** (bool reuse)

  Sets the reuse address flag, SO_REUSEADDR.
- virtual int **getSendBufferSize** () **const**

  Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.
- virtual void **setSendBufferSize** (int size)

  Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.
- virtual int **getSoTimeout** () **const**

  Gets the timeout for socket operations, SO_TIMEOUT.
- virtual void **setSoTimeout** (int timeout)

  Sets the timeout for socket operations, SO_TIMEOUT.
- virtual bool **getTcpNoDelay** () **const**

  Gets the Status of the TCP_NODELAY setting for this socket.
- virtual void **setTcpNoDelay** (bool value)

  *Sets the Status of the TCP_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 1900).*
- virtual int **getTrafficClass** () **const**

  *Gets the Traffic Class setting for this **Socket** (p. 1900), sometimes referred to as Type of Service setting.*
- virtual void **setTrafficClass** (int value)

  *Gets the Traffic Class setting for this **Socket** (p. 1900), sometimes referred to as Type of Service setting.*
- virtual bool **getOOBInline** () **const**

  Gets the value of the OOBINLINE for this socket.
- virtual void **setOOBInline** (bool value)

  Sets the value of the OOBINLINE for this socket, by default this option is disabled.
- virtual void **sendUrgentData** (int data)

  *Sends on byte of urgent data to the **Socket** (p. 1900).*
- virtual std::string **toString** () **const**

Static Public Member Functions

- static void **setSocketImplFactory** (**SocketImplFactory** *factory)

  *Sets the instance of a **SocketImplFactory** (p. 1928) that the **Socket** (p. 1900) class should use when new instances of this class are created.*

Protected Member Functions

- void **accepted** ()
- void **initSocketImpl** (**const** std::string &address, int port, **const InetAddress** *localAddress, int localPort)
- void **checkClosed** () **const**
- void **ensureCreated** () **const**

Protected Attributes

- **SocketImpl** * **impl**

Friends

- class **ServerSocket**

6.487.1 Detailed Description

Since

1.0

6.487.2 Constructor & Destructor Documentation

6.487.2.1 decaf::net::Socket::Socket ( )

Creates an unconnected **Socket** (p. 1900) using the set **SocketImplFactory** (p. 1928) or if non is set than the default SockImpl type is created.

6.487.2.2 decaf::net::Socket::Socket (SocketImpl * *impl*)

Creates a **Socket** (p. 1900) wrapping the provided **SocketImpl** (p. 1921) instance, this **Socket** (p. 1900) is considered unconnected.

The **Socket** (p. 1900) class takes ownership of this **SocketImpl** (p. 1921) pointer and will delete it when the **Socket** (p. 1900) class is destroyed.

Parameters

impl	The **SocketImpl** (p. 1921) instance to wrap.

Exceptions

NullPointerException	if the passed **SocketImpl** (p. 1921) is Null.

6.487.2.3 decaf::net::Socket::Socket (const InetAddress * *address,* int *port*)

Creates a new **Socket** (p. 1900) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 1928) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 1900) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

address	The address to connect to.
port	The port number to connect to [0...65535]

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
NullPointerException	if the **InetAddress** (p. 1113) instance in NULL.
IllegalArgumentException	if the port if not in range [0...65535]

6.487.2.4 decaf::net::Socket::Socket (const InetAddress * *address*, int *port*, const InetAddress * *localAddress*, int *localPort*)

Creates a new **Socket** (p. 1900) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 1928) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 1900) implementation is used. The **Socket** (p. 1900) will also bind to the local address and port specified.

Parameters

address	The address to connect to.
port	The port number to connect to [0...65535]
localAddress	The IP address on the local machine to bind to.
localPort	The port on the local machine to bind to.

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
NullPointerException	if the **InetAddress** (p. 1113) instance in NULL.
IllegalArgumentException	if the port if not in range [0...65535]

6.487.2.5 decaf::net::Socket::Socket (const std::string & *host*, int *port*)

Creates a new **Socket** (p. 1900) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 1928) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 1900) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

host	The host name or IP address to connect to, empty string means loopback.
port	The port number to connect to [0...65535]

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
IllegalArgumentException	if the port if not in range [0...65535]

6.487.2.6 decaf::net::Socket::Socket (const std::string & *host,* int *port,* const InetAddress * *localAddress,* int *localPort*)

Creates a new **Socket** (p. 1900) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 1928) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 1900) implementation is used.

If the host parameter is empty then the loop back address is used.

Parameters

host	The host name or IP address to connect to, empty string means loopback.
port	The port number to connect to [0...65535]
localAddress	The IP address on the local machine to bind to.
localPort	The port on the local machine to bind to.

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
IllegalArgumentException	if the port if not in range [0...65535]

6.487.2.7 virtual decaf::net::Socket::~Socket ( ) [virtual]

6.487.3 Member Function Documentation

6.487.3.1 void decaf::net::Socket::accepted ( ) [protected]

6.487.3.2 virtual void decaf::net::Socket::bind (const std::string & *ipaddress,* int *port*) [virtual]

Binds this **Socket** (p. 1900) to the given local address and port.

If the **SocketAddress** (p. 1913) value is NULL then the **Socket** (p. 1900) will be bound to an available local address and port.

Parameters

ipaddress	The local address and port to bind the socket to.
port	The port on the local machine to bind to.

Exceptions

IOException	if an error occurs during the bind operation.
IllegalArgumentException	if the **Socket** (p. 1900) can't process the subclass of **SocketAddress** (p. 1913) that has been provided.

6.487.3.3 void decaf::net::Socket::checkClosed ( ) const [protected]

6.487.3.4 virtual void decaf::net::Socket::close ( ) [virtual]

Closes the **Socket** (p. 1900).

Once closed a **Socket** (p. 1900) cannot be connected or otherwise operated upon, a new **Socket** (p. 1900) instance must be created.

Exceptions

IOException	if an I/O error occurs while closing the **Socket** (p. 1900).

Implements **decaf::io::Closeable** (p. 633).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1565).

6.487.3.5 virtual void decaf::net::Socket::connect (const std::string & *host,* int *port*) [virtual]

Connects to the specified destination.

Parameters

host	The host name or IP address of the remote host to connect to.
port	The port on the remote host to connect to.

Exceptions

IOException	Thrown if a failure occurred in the connect.
IllegalArguementException	if the timeout value is negative or the endpoint is invalid.

6.487.3.6 virtual void decaf::net::Socket::connect (const std::string & *host,* int *port,* int *timeout*) [virtual]

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 1932) is thrown. A timeout value of zero is treated as an inifinite timeout.

Parameters

host	The host name or IP address of the remote host to connect to.
port	The port on the remote host to connect to.
timeout	The number of Milliseconds to wait before treating the connection as failed.

Exceptions

IOException	Thrown if a failure occurred in the connect.
SocketTimeoutException *(p. 1932)*	if the timeout for connection is exceeded.
IllegalArguementException	if the timeout value is negative or the endpoint is invalid.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1565).

6.487.3.7 void decaf::net::Socket::ensureCreated ( ) const [protected]

6.487.3.8 std::string decaf::net::Socket::getInetAddress ( ) const

Returns the address to which the socket is connected.

Returns

the remote IP address to which this socket is connected, or null if the socket is not connected.

6.487.3.9 virtual decaf::io::InputStream* decaf::net::Socket::getInputStream ( ) [virtual]

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 1900) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 1900).

Returns

The InputStream for this socket.

Exceptions

IOException	if an error occurs during creation of the InputStream, also if the **Socket** (p. 1900) is not connected or the input has been shutdown previously.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1566).

6.487.3.10 virtual bool decaf::net::Socket::getKeepAlive ( ) const [virtual]

Gets the keep alive flag for this socket, SO_KEEPALIVE.

Returns

true if keep alive is enabled for this socket.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.487.3.11 std::string decaf::net::Socket::getLocalAddress ( ) const

Gets the local address to which the socket is bound.

Returns

the local address to which the socket is bound or InetAddress.anyLocalAddress() if the socket is not bound yet.

6.487.3.12 int decaf::net::Socket::getLocalPort ( ) const

Gets the local port the socket is bound to.

Returns

the local port the socket was bound to, or -1 if the socket is not bound.

6.487.3.13 virtual bool decaf::net::Socket::getOOBInline ( ) const [virtual]

Gets the value of the OOBINLINE for this socket.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException *(p. 1915)*	if an error is encountered while performing this operation.

6.487.3.14 virtual decaf::io::OutputStream* decaf::net::Socket::getOutputStream ( ) [virtual]

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 1900) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 1900) will also close the underlying **Socket** (p. 1900).

Returns

the OutputStream for this socket.

Exceptions

IOException	if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 1900) is closed or the output has been shutdown previously.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1566).

6.487.3.15 int decaf::net::Socket::getPort ( ) const

Gets the on the remote host this **Socket** (p. 1900) is connected to.

Returns

the port on the remote host the socket is connected to, or 0 if not connected.

6.487.3.16 virtual int decaf::net::Socket::getReceiveBufferSize ( ) const [virtual]

Gets the receive buffer size for this socket, SO_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

Returns

the receive buffer size in bytes.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.487.3.17 virtual bool decaf::net::Socket::getReuseAddress ( ) const [virtual]

Gets the reuse address flag, SO_REUSEADDR.

Returns

True if the address can be reused.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.487.3.18 virtual int decaf::net::Socket::getSendBufferSize ( ) const [virtual]

Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

Returns

the size in bytes of the send buffer.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.487.3.19 virtual int decaf::net::Socket::getSoLinger ( ) const [virtual]

Gets the linger time for the socket, SO_LINGER.

A return value of -1 indicates that the option is disabled.

Returns

The linger time in seconds.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.487.3.20 virtual int decaf::net::Socket::getSoTimeout ( ) const [virtual]

Gets the timeout for socket operations, SO_TIMEOUT.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException *(p. 1915)*	Thrown if unable to retrieve the information.

6.487.3.21 virtual bool decaf::net::Socket::getTcpNoDelay ( ) const [virtual]

Gets the Status of the TCP_NODELAY setting for this socket.

Returns

true if TCP_NODELAY is enabled for the socket.

Exceptions

SocketException *(p. 1915)*	Thrown if unable to set the information.

6.487.3.22 virtual int decaf::net::Socket::getTrafficClass ( ) const [virtual]

Gets the Traffic Class setting for this **Socket** (p. 1900), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 1900) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

Returns

the bitset result of querying the traffic class setting.

Exceptions

SocketException *(p. 1915)*	if an error is encountered while performing this operation.

6.487.3.23 void decaf::net::Socket::initSocketImpl (const std::string & *address,* int *port,* const InetAddress * *localAddress,* int *localPort*) [protected]

6.487.3.24 bool decaf::net::Socket::isBound ( ) const [inline]

Returns

true if this **Socket** (p. 1900) has been bound to a Local address.

6.487.3.25 bool decaf::net::Socket::isClosed ( ) const [inline]

Returns

true if the **Socket** (p. 1900) has been closed.

6.487.3.26 bool decaf::net::Socket::isConnected ( ) const [inline]

Indicates whether or not this socket is connected to am end point.

Returns

true if connected, false otherwise.

6.487.3.27 bool decaf::net::Socket::isInputShutdown ( ) const [inline]

Returns

true if input on this **Socket** (p. 1900) has been shutdown.

6.487.3.28 bool decaf::net::Socket::isOutputShutdown ( ) const [inline]

Returns

true if output on this **Socket** (p. 1900) has been shutdown.

6.487.3.29 virtual void decaf::net::Socket::sendUrgentData (int *data*) [virtual]

Sends on byte of urgent data to the **Socket** (p. 1900).

Parameters

data	The value to write as urgent data, only the lower eight bits are sent.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1568).

6.487.3.30 virtual void decaf::net::Socket::setKeepAlive (bool *keepAlive*) [virtual]

Enables/disables the keep alive flag for this socket, SO_KEEPALIVE.

Parameters

keepAlive	If true, enables the flag.

Exceptions

SocketException *(p. 1915)*	if the operation fails.

6.487.3.31 virtual void decaf::net::Socket::setOOBInline (bool *value*) [virtual]

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the **Socket** (p. 1900)'s InputStream, no notification is give.

Returns

true if OOBINLINE is enabled, false otherwise.

Exceptions

SocketException *(p. 1915)*	if an error is encountered while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1569).

6.487.3.32 virtual void decaf::net::Socket::setReceiveBufferSize (int *size*) [virtual]

Sets the receive buffer size for this socket, SO_RCVBUF.

Parameters

size	Number of bytes to set the receive buffer to.

Exceptions

SocketException *(*p. *1915)*	if the operation fails.
IllegalArgumentException	if the value is zero or negative.

6.487.3.33 virtual void decaf::net::Socket::setReuseAddress (bool *reuse*) [virtual]

Sets the reuse address flag, SO_REUSEADDR.

Parameters

reuse	If true, sets the flag.

Exceptions

SocketException *(*p. *1915)*	if the operation fails.

6.487.3.34 virtual void decaf::net::Socket::setSendBufferSize (int *size*) [virtual]

Gets the send buffer size for this socket, SO_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

Parameters

size	The number of bytes to set the send buffer to, must be larger than zero.

Exceptions

SocketException *(*p. *1915)*	if the operation fails.
IllegalArgumentException	if the value is zero or negative.

6.487.3.35 static void decaf::net::Socket::setSocketImplFactory (SocketImplFactory * *factory*) [static]

Sets the instance of a **SocketImplFactory** (p. 1928) that the **Socket** (p. 1900) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

Parameters

factory	The instance of a **SocketImplFactory** (p. 1928) to use when new **Socket** (p. 1900) objects are created.

Exceptions

IOException	if an I/O error occurs while performing this operation.
SocketException *(*p. *1915)*	if this method has already been called with a valid factory.

6.487.3.36 virtual void decaf::net::Socket::setSoLinger (bool *state,* int *timeout*) [virtual]

Sets the linger time (SO_LINGER) using a specified time value, this limits of this value are platform specific.

Parameters

state	The state of SO_LINGER, true is on.
timeout	The linger time in seconds, must be non-negative.

Exceptions

SocketException *(*p. *1915)*	if the operation fails.
IllegalArgumentException	if state is true and timeout is negative.

6.487.3.37 virtual void decaf::net::Socket::setSoTimeout (int *timeout*) [virtual]

Sets the timeout for socket operations, SO_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

Parameters

timeout	The timeout in milliseconds for socket operations.

Exceptions

SocketException *(*p. *1915)*	Thrown if unable to set the information.
IllegalArgumentException	if the timeout value is negative.

6.487.3.38 virtual void decaf::net::Socket::setTcpNoDelay (bool *value*) [virtual]

Sets the Status of the TCP_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 1900).

Parameters

value	The setting for the socket's TCP_NODELAY option, true to enable.

Exceptions

SocketException *(*p. *1915)*	Thrown if unable to set the information.

6.487.3.39 virtual void decaf::net::Socket::setTrafficClass (int *value*) [virtual]

Gets the Traffic Class setting for this **Socket** (p. 1900), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 1900) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

Parameters

value	The integer value representing the traffic class setting bitset.

Exceptions

SocketException *(p. 1915)*	if an error is encountered while performing this operation.
IllegalArgumentException	if the value is not in the range [0..255].

6.487.3.40 virtual void decaf::net::Socket::shutdownInput ( ) [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1570).

6.487.3.41 virtual void decaf::net::Socket::shutdownOutput ( ) [virtual]

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OuputStream::write will throw an IOException.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1570).

6.487.3.42 virtual std::string decaf::net::Socket::toString ( ) const [virtual]

Returns

a string representing this **Socket** (p. 1900).

6.487.4 Friends And Related Function Documentation

6.487.4.1 friend class ServerSocket [friend]

6.487.5 Field Documentation

6.487.5.1 SocketImpl* decaf::net::Socket::impl [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**Socket.h**

6.488 decaf::net::SocketAddress Class Reference

Base class for protocol specific **Socket** (p. 1900) addresses.

`#include <src/main/decaf/net/SocketAddress.h>`

Inheritance diagram for decaf::net::SocketAddress:

Public Member Functions

- virtual ~**SocketAddress** ()

6.488.1 Detailed Description

Base class for protocol specific **Socket** (p. 1900) addresses.

These classes provide an immutable address object that is used by the **Socket** (p. 1900) classes.

Since

1.0

6.488.2 Constructor & Destructor Documentation

6.488.2.1 virtual decaf::net::SocketAddress::~SocketAddress ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketAddress.h**

6.489 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.

6.489.1 Detailed Description

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

6.489.2 Member Function Documentation

6.489.2.1 static int decaf::net::SocketError::getErrorCode ( ) [static]

Gets the last error appropriate for the platform.

6.489.2.2 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**

6.490 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 ()

6.490.1 Detailed Description

Exception for errors when manipulating sockets.

6.490.2 Constructor & Destructor Documentation

6.490.2.1 decaf::net::SocketException::SocketException ( ) throw () [inline]

6.490.2.2 decaf::net::SocketException::SocketException (const lang::Exception & *ex*) throw () [inline]

6.490.2.3 decaf::net::SocketException::SocketException (const SocketException & *ex*) throw () [inline]

6.490.2.4 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

6.490.2.5 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.

6.490.2.6 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

6.490.2.7 virtual decaf::net::SocketException::~SocketException ( ) throw () [inline, virtual]

6.490.3 Member Function Documentation

6.490.3.1 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. 1200).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 1574), **decaf::net::BindException** (p. 413), **decaf::net::ConnectException** (p. 724), **decaf::net::NoRouteToHostException** (p. 1534), and **decaf::net::PortUnreachableException** (p. 1634).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketException.h**

6.491 decaf::net::SocketFactory Class Reference

The **SocketFactory** (p. 1916) is used to create **Socket** (p. 1900) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

`#include <src/main/decaf/net/SocketFactory.h>`

Inheritance diagram for decaf::net::SocketFactory:

Public Member Functions

- virtual ~**SocketFactory** ()
- virtual **Socket** * **createSocket** ()

  *Creates an unconnected **Socket** (p. 1900) object.*
- virtual **Socket** * **createSocket** (**const InetAddress** *host, int port)=0

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*
- virtual **Socket** * **createSocket** (**const InetAddress** *host, int port, **const InetAddress** *ifAddress, int localPort)=0

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*
- virtual **Socket** * **createSocket** (**const** std::string &name, int port)=0

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*
- virtual **Socket** * **createSocket** (**const** std::string &name, int port, **const InetAddress** *ifAddress, int localPort)=0

  *Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).*

Static Public Member Functions

- static **SocketFactory** * **getDefault** ()

  *Returns an pointer to the default **SocketFactory** (p. 1916) for this Application, there is only one default **SocketFactory** (p. 1916) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 1916) class and in not to be deleted by the caller.*

Protected Member Functions

- **SocketFactory** ()

6.491.1 Detailed Description

The **SocketFactory** (p. 1916) is used to create **Socket** (p. 1900) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

See also

decaf.net.Socket (p. 1900)

Since

1.0

6.491.2 Constructor & Destructor Documentation

6.491.2.1 decaf::net::SocketFactory::SocketFactory ( ) [protected]

6.491.2.2 virtual decaf::net::SocketFactory::~SocketFactory ( ) [virtual]

6.491.3 Member Function Documentation

6.491.3.1 virtual Socket* decaf::net::SocketFactory::createSocket ( ) [virtual]

Creates an unconnected **Socket** (p. 1900) object.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if the **Socket** (p. 1900) cannot be created.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1577), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 909), and **decaf::internal::net::DefaultSocketFactory** (p. 899).

6.491.3.2 virtual Socket* decaf::net::SocketFactory::createSocket (const InetAddress * *host,* int *port*) [pure virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1577), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 909), and **decaf::internal::net::DefaultSocketFactory** (p. 899).

6.491.3.3 virtual Socket* decaf::net::SocketFactory::createSocket (const InetAddress * *host,* int *port,* const InetAddress * *ifAddress,* int *localPort*) [pure virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

The **Socket** (p. 1900) will be bound to the specified local address and port.

Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1577), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 909), and **decaf::internal::net::DefaultSocketFactory** (p. 900).

6.491.3.4 virtual Socket* decaf::net::SocketFactory::createSocket (const std::string & *name,* int *port*) [pure virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1578), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 910), and **decaf::internal::net::DefaultSocketFactory** (p. 900).

6.491.3.5 virtual Socket* decaf::net::SocketFactory::createSocket (const std::string & *name,* int *port,* const InetAddress * *ifAddress,* int *localPort*) [pure virtual]

Creates a new **Socket** (p. 1900) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 1916).

Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the **Socket** (p. 1900) to.
localPort	The local port to bind the **Socket** (p. 1900) to.

Returns

a new **Socket** (p. 1900) object, caller must free this object when done.

Exceptions

IOException	if an I/O error occurs while creating the **Socket** (p. 1900) object.
UnknownHostException *(p. 2181)*	if the host name is not known.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1578), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 910), and **decaf::internal::net::DefaultSocketFactory** (p. 901).

6.491.3.6 static SocketFactory* decaf::net::SocketFactory::getDefault ( ) [static]

Returns an pointer to the default **SocketFactory** (p. 1916) for this Application, there is only one default **SocketFactory** (p. 1916) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 1916) class and in not to be deleted by the caller.

Returns

pointer to the applications default **SocketFactory** (p. 1916).

Exceptions

SocketException *(p. 1915)*	if an error occurs while getting the default instance.

Reimplemented in **decaf::net::ssl::SSLSocketFactory** (p. 1955).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketFactory.h**

6.492 decaf::internal::net::SocketFileDescriptor Class Reference

File Descriptor type used internally by Decaf Socket objects.

`#include <src/main/decaf/internal/net/SocketFileDescriptor.h>`

Inheritance diagram for decaf::internal::net::SocketFileDescriptor:

Public Member Functions

- **SocketFileDescriptor** (long value)
- virtual ~**SocketFileDescriptor** ()
- long **getValue** () **const**

  Gets the OS Level FileDescriptor.

6.492.1 Detailed Description

File Descriptor type used internally by Decaf Socket objects.

Since

1.0

6.492.2 Constructor & Destructor Documentation

6.492.2.1 decaf::internal::net::SocketFileDescriptor::SocketFileDescriptor (long *value*)

6.492.2.2 virtual decaf::internal::net::SocketFileDescriptor::~SocketFileDescriptor ( ) [virtual]

6.492.3 Member Function Documentation

6.492.3.1 long decaf::internal::net::SocketFileDescriptor::getValue ( ) const

Gets the OS Level FileDescriptor.

Returns

a FileDescriptor value.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**SocketFileDescriptor.h**

6.493 decaf::net::SocketImpl Class Reference

Acts as a base class for all physical **Socket** (p. 1900) implementations.

`#include <src/main/decaf/net/SocketImpl.h>`

Inheritance diagram for decaf::net::SocketImpl:

Public Member Functions

- **SocketImpl** ()
- virtual ∼**SocketImpl** ()
- virtual void **create** ()=0

  *Creates the underlying platform **Socket** (p. 1900) data structures which allows for **Socket** (p. 1900) options to be applied.*
- virtual void **accept** (**SocketImpl** ∗socket)=0

  *Accepts a new connection on the given **Socket** (p. 1900).*
- virtual void **connect** (**const** std::string &hostname, int **port**, int timeout)=0

  Connects this socket to the given host and port.
- virtual void **bind** (**const** std::string &ipaddress, int **port**)=0

  *Binds this **Socket** (p. 1900) instance to the local ip address and port number given.*
- virtual void **listen** (int backlog)=0

  Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.
- virtual **decaf::io::InputStream** ∗ **getInputStream** ()=0

  *Gets the InputStream linked to this **Socket** (p. 1900).*
- virtual **decaf::io::OutputStream** ∗ **getOutputStream** ()=0

  *Gets the OutputStream linked to this **Socket** (p. 1900).*
- virtual int **available** ()=0

  *Gets the number of bytes that can be read from the **Socket** (p. 1900) without blocking.*
- virtual void **close** ()=0

  Closes the socket, terminating any blocked reads or writes.
- virtual void **shutdownInput** ()=0

  Places the input stream for this socket at "end of stream".
- virtual void **shutdownOutput** ()=0

  Disables the output stream for this socket.
- virtual int **getOption** (int option) **const** =0

  *Gets the specified **Socket** (p. 1900) option.*

- virtual void **setOption** (int option, int value)=0

  *Sets the specified option on the **Socket** (p. 1900) if supported.*
- int **getPort** () **const**

  Gets the port that this socket has been assigned.
- int **getLocalPort** () **const**

  *Gets the value of this **SocketImpl** (p. 1921)'s local port field.*
- std::string **getInetAddress** () **const**

  *Gets the value of this **SocketImpl** (p. 1921)'s address field.*
- **const decaf::io::FileDescriptor** * **getFileDescriptor** () **const**

  *Gets the FileDescriptor for this **Socket** (p. 1900), the Object is owned by this **Socket** (p. 1900) and should not be deleted by the caller.*
- virtual std::string **getLocalAddress** () **const** =0

  *Gets the value of the local Inet address the **Socket** (p. 1900) is bound to if bound, otherwise return the **InetAddress** (p. 1113) ANY value "0.0.0.0".*
- std::string **toString** () **const**

  *Returns a string containing the address and port of this **Socket** (p. 1900) instance.*
- virtual bool **supportsUrgentData** () **const**
- virtual void **sendUrgentData** (int data)

  *Sends on byte of urgent data to the **Socket** (p. 1900).*

Protected Attributes

- int **port**

  *The remote port that this **Socket** (p. 1900) is connected to.*
- int **localPort**

  *The port on the Local Machine that this **Socket** (p. 1900) is Bound to.*
- std::string **address**

  *The Remote Address that the **Socket** (p. 1900) is connected to.*
- **io::FileDescriptor** * **fd**

  *The File Descriptor for this **Socket** (p. 1900).*

6.493.1 Detailed Description

Acts as a base class for all physical **Socket** (p. 1900) implementations.

Since

1.0

6.493.2 Constructor & Destructor Documentation

6.493.2.1 decaf::net::SocketImpl::SocketImpl ( )

6.493.2.2 virtual decaf::net::SocketImpl::~SocketImpl ( ) [virtual]

6.493.3 Member Function Documentation

6.493.3.1 virtual void decaf::net::SocketImpl::accept (SocketImpl * *socket*) [pure virtual]

Accepts a new connection on the given **Socket** (p. 1900).

Parameters

socket	The accepted connection.

Exceptions

IOException	if an I/O error occurs while attempting this operation.
SocketException *(p. 1915)*	if an error occurs while performing an Accept on the socket.
SocketTimeoutException *(p. 1932)*	if the accept call times out due to SO_TIMEOUT being set.

6.493.3.2 virtual int decaf::net::SocketImpl::available ( ) [pure virtual]

Gets the number of bytes that can be read from the **Socket** (p. 1900) without blocking.

Returns

the number of bytes that can be read from the **Socket** (p. 1900) without blocking.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2077).

6.493.3.3 virtual void decaf::net::SocketImpl::bind (const std::string & *ipaddress*, int *port*) [pure virtual]

Binds this **Socket** (p. 1900) instance to the local ip address and port number given.

Parameters

ipaddress	The address of local ip to bind to.
port	The port number on the host to bind to.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2078).

6.493.3.4 virtual void decaf::net::SocketImpl::close ( ) [pure virtual]

Closes the socket, terminating any blocked reads or writes.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2078).

6.493.3.5 virtual void decaf::net::SocketImpl::connect (const std::string & *hostname*, int *port*, int *timeout*) [pure virtual]

Connects this socket to the given host and port.

Parameters

hostname	The name of the host to connect to, or IP address.
port	The port number on the host to connect to.
timeout	Time in milliseconds to wait for a connection, 0 indicates forever.

Exceptions

IOException	if an I/O error occurs while attempting this operation.
SocketTimeoutException *(p. 1932)*	if the connect call times out due to timeout being set.
IllegalArguementException	if a parameter has an illegal value.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2078).

6.493.3.6 virtual void decaf::net::SocketImpl::create ( ) [pure virtual]

Creates the underlying platform **Socket** (p. 1900) data structures which allows for **Socket** (p. 1900) options to be applied.

The created socket is in an unconnected state.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2079).

6.493.3.7 const decaf::io::FileDescriptor* decaf::net::SocketImpl::getFileDescriptor ( ) const [inline]

Gets the FileDescriptor for this **Socket** (p. 1900), the Object is owned by this **Socket** (p. 1900) and should not be deleted by the caller.

Returns

a pointer to this **Socket** (p. 1900)'s FileDescriptor object.

6.493.3.8 std::string decaf::net::SocketImpl::getInetAddress ( ) const [inline]

Gets the value of this **SocketImpl** (p. 1921)'s address field.

Returns

the value of the address field.

6.493.3.9 virtual decaf::io::InputStream* decaf::net::SocketImpl::getInputStream ( ) [pure virtual]

Gets the InputStream linked to this **Socket** (p. 1900).

Returns

an InputStream pointer owned by the **Socket** (p. 1900) object.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2079).

6.493.3.10 virtual std::string decaf::net::SocketImpl::getLocalAddress ( ) const [pure virtual]

Gets the value of the local Inet address the **Socket** (p. 1900) is bound to if bound, otherwise return the **InetAddress** (p. 1113) ANY value "0.0.0.0".

Returns

the local address bound to, or ANY.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2079).

6.493.3.11 int decaf::net::SocketImpl::getLocalPort ( ) const [inline]

Gets the value of this **SocketImpl** (p. 1921)'s local port field.

Returns

the value of localPort.

6.493.3.12 virtual int decaf::net::SocketImpl::getOption (int *option*) const [pure virtual]

Gets the specified **Socket** (p. 1900) option.

Parameters

option	The **Socket** (p. 1900) options whose value is to be retrieved.

Returns

the value of the given socket option.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2079).

6.493.3.13 virtual decaf::io::OutputStream* decaf::net::SocketImpl::getOutputStream ( ) [pure virtual]

Gets the OutputStream linked to this **Socket** (p. 1900).

Returns

an OutputStream pointer owned by the **Socket** (p. 1900) object.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2080).

6.493.3.14 int decaf::net::SocketImpl::getPort ( ) const [inline]

Gets the port that this socket has been assigned.

Returns

the **Socket** (p. 1900)'s port number.

6.493.3.15 virtual void decaf::net::SocketImpl::listen (int *backlog*) [pure virtual]

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

Parameters

backlog	The maximum length of the connection queue.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2080).

6.493.3.16 virtual void decaf::net::SocketImpl::sendUrgentData (int *data*) [virtual]

Sends on byte of urgent data to the **Socket** (p. 1900).

Parameters

data	The value to write as urgent data, only the lower eight bits are sent.

Exceptions

IOException	if an I/O error occurs while performing this operation.

6.493.3.17 virtual void decaf::net::SocketImpl::setOption (int *option,* int *value*) [pure virtual]

Sets the specified option on the **Socket** (p. 1900) if supported.

Parameters

option	The **Socket** (p. 1900) option to set.
value	The value of the socket option to apply to the socket.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2081).

6.493.3.18 virtual void decaf::net::SocketImpl::shutdownInput ( ) [pure virtual]

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 1926) on the socket, the stream will return EOF.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2081).

6.493.3.19 virtual void decaf::net::SocketImpl::shutdownOutput ( ) [pure virtual]

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 1927) on the socket, the stream will throw an IOException.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 2081).

6.493.3.20 virtual bool decaf::net::SocketImpl::supportsUrgentData ( ) const [inline, virtual]

Returns

true if this **SocketImpl** (p. 1921) supports sending Urgent Data. The default implementation always returns false.

6.493.3.21 std::string decaf::net::SocketImpl::toString ( ) const

Returns a string containing the address and port of this **Socket** (p. 1900) instance.

Returns

a string containing the address and port of this socket.

6.493.4 Field Documentation

6.493.4.1 std::string decaf::net::SocketImpl::address [protected]

The Remote Address that the **Socket** (p. 1900) is connected to.

6.493.4.2 io::FileDescriptor* decaf::net::SocketImpl::fd [protected]

The File Descriptor for this **Socket** (p. 1900).

6.493.4.3 int decaf::net::SocketImpl::localPort [protected]

The port on the Local Machine that this **Socket** (p. 1900) is Bound to.

6.493.4.4 int decaf::net::SocketImpl::port [protected]

The remote port that this **Socket** (p. 1900) is connected to.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketImpl.h**

6.494 decaf::net::SocketImplFactory Class Reference

Factory class interface for a Factory that creates ScoketImpl objects.

`#include <src/main/decaf/net/SocketImplFactory.h>`

Public Member Functions

- virtual ~**SocketImplFactory** ()
- virtual **SocketImpl** * **createSocketImpl** ()=0

  *Creates a new SokcetImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 1921).*

6.494.1 Detailed Description

Factory class interface for a Factory that creates ScoketImpl objects.

These factories can be used to create various types of Sockets, e.g. Streaming, Multicast, SSL, or platform specific variations of these types.

See also

decaf::net::Socket (p. 1900)
decaf::net::ServerSocket (p. 1816)

Since

1.0

6.494.2 Constructor & Destructor Documentation

6.494.2.1 virtual decaf::net::SocketImplFactory::~SocketImplFactory ( ) [inline, virtual]

6.494.3 Member Function Documentation

6.494.3.1 virtual SocketImpl* decaf::net::SocketImplFactory::createSocketImpl ( ) [pure virtual]

Creates a new SokcetImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 1921).

Returns

new **SocketImpl** (p. 1921) instance that is owned by the caller.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketImplFactory.h**

6.495 decaf::net::SocketOptions Class Reference

`#include <src/main/decaf/net/SocketOptions.h>`

Inheritance diagram for decaf::net::SocketOptions:

Public Member Functions

- virtual ~**SocketOptions** ()

Static Public Attributes

- static **const** int **SOCKET_OPTION_TCP_NODELAY**

  Disable Nagle's algorithm for this connection.
- static **const** int **SOCKET_OPTION_BINDADDR**

  Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).
- static **const** int **SOCKET_OPTION_REUSEADDR**

  Sets SO_REUSEADDR for a socket.
- static **const** int **SOCKET_OPTION_BROADCAST**

  Sets SO_BROADCAST for a socket.
- static **const** int **SOCKET_OPTION_IP_MULTICAST_IF**

  Set which outgoing interface on which to send multicast packets.
- static **const** int **SOCKET_OPTION_IP_MULTICAST_IF2**

  Same as above.
- static **const** int **SOCKET_OPTION_IP_MULTICAST_LOOP**

  This option enables or disables local loopback of multicast datagrams.
- static **const** int **SOCKET_OPTION_IP_TOS**

  This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.
- static **const** int **SOCKET_OPTION_LINGER**

  Specify a linger-on-close timeout.
- static **const** int **SOCKET_OPTION_TIMEOUT**

  *Set a timeout on blocking **Socket** (p. 1900) operations.*
- static **const** int **SOCKET_OPTION_SNDBUF**

  Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.
- static **const** int **SOCKET_OPTION_RCVBUF**

  Set a hint the size of the underlying buffers used by the platform for incoming network I/O.
- static **const** int **SOCKET_OPTION_KEEPALIVE**

  When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.
- static **const** int **SOCKET_OPTION_OOBINLINE**

  When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

6.495.1 Detailed Description

Since

1.0

6.495.2 Constructor & Destructor Documentation

6.495.2.1 virtual decaf::net::SocketOptions::~SocketOptions ( ) [virtual]

6.495.3 Field Documentation

6.495.3.1 const int decaf::net::SocketOptions::SOCKET_OPTION_BINDADDR [static]

Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).

The default local address of a socket is INADDR_ANY, meaning any local address on a multi-homed host. A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a **ServerSocket** (p. 1816) or DatagramSocket), or to specify its return address to the peer (for a **Socket** (p. 1900) or DatagramSocket). The parameter of this option is an **InetAddress** (p. 1113).

6.495.3.2 const int decaf::net::SocketOptions::SOCKET_OPTION_BROADCAST [static]

Sets SO_BROADCAST for a socket.

This option enables and disables the ability of the process to send broadcast messages. It is supported for only datagram sockets and only on networks that support the concept of a broadcast message (e.g. Ethernet, token ring, etc.), and it is set by default for DatagramSockets.

6.495.3.3 const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_IF [static]

Set which outgoing interface on which to send multicast packets.

Useful on hosts with multiple network interfaces, where applications want to use other than the system default. Takes/returns an **InetAddress** (p. 1113).

Valid for Multicast: DatagramSocketImpl.

6.495.3.4 const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_IF2 [static]

Same as above.

This option is introduced so that the behaviour with IP_MULTICAST_IF will be kept the same as before, while this new option can support setting outgoing interfaces with either IPv4 and IPv6 addresses.

6.495.3.5 const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_LOOP [static]

This option enables or disables local loopback of multicast datagrams.

This option is enabled by default for Multicast Sockets.

6.495.3.6 const int decaf::net::SocketOptions::SOCKET_OPTION_IP_TOS [static]

This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.

6.495.3.7 const int decaf::net::SocketOptions::SOCKET_OPTION_KEEPALIVE [static]

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.

This probe is a TCP segment to which the peer must respond. One of three responses is expected: 1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity. 2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed. 3. There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host crashes.

Valid only for TCP socket: **SocketImpl** (p. 1921)

6.495.3.8 const int decaf::net::SocketOptions::SOCKET_OPTION_LINGER [static]

Specify a linger-on-close timeout.

This option disables/enables immediate return from a close() of a TCP **Socket** (p. 1900). Enabling this option with a non-zero Integer timeout means that a close() will block pending the transmission and acknowledgment of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully, with a TCP RST. Enabling the option with a timeout of zero does a forceful close immediately. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

Valid only for TCP: **SocketImpl** (p. 1921)

6.495.3.9 const int decaf::net::SocketOptions::SOCKET_OPTION_OOBINLINE [static]

When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

When the option is disabled (which is the default) urgent data is silently discarded.

6.495.3.10 const int decaf::net::SocketOptions::SOCKET_OPTION_RCVBUF [static]

Set a hint the size of the underlying buffers used by the platform for incoming network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be received over the socket. When used in get, this must return the size of the buffer actually used by the platform when receiving in data on this socket. Valid for all sockets: **SocketImpl** (p. 1921), DatagramSocketImpl.

6.495.3.11 const int decaf::net::SocketOptions::SOCKET_OPTION_REUSEADDR [static]

Sets SO_REUSEADDR for a socket.

This is used only for MulticastSockets in decaf, and it is set by default for MulticastSockets.

6.495.3.12 const int decaf::net::SocketOptions::SOCKET_OPTION_SNDBUF [static]

Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be sent over the socket. When used in get, this must return the size of the buffer actually used by the platform when sending out data on this socket. Valid for all sockets: **SocketImpl** (p. 1921), DatagramSocketImpl

6.495.3.13 const int decaf::net::SocketOptions::SOCKET_OPTION_TCP_NODELAY [static]

Disable Nagle's algorithm for this connection.

Written data to the network is not buffered pending acknowledgment of previously written data. Valid for TCP sockets.

6.495.3.14 const int decaf::net::SocketOptions::SOCKET_OPTION_TIMEOUT [static]

Set a timeout on blocking **Socket** (p. 1900) operations.

The option must be set prior to entering a blocking operation to take effect.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketOptions.h**

6.496 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 ()

6.496.1 Constructor & Destructor Documentation

6.496.1.1 decaf::net::SocketTimeoutException::SocketTimeoutException ( ) throw () [inline]

Default Constructor.

6.496.1.2 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

6.496.1.3 decaf::net::SocketTimeoutException::SocketTimeoutException (const SocketTimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.496.1.4 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

6.496.1.5 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.

6.496.1.6 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

6.496.1.7 virtual decaf::net::SocketTimeoutException::~SocketTimeoutException ( ) throw () [inline, virtual]

6.496.2 Member Function Documentation

6.496.2.1 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. 1189).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketTimeoutException.h**

6.497 decaf::net::ssl::SSLContext Class Reference

Represents on implementation of the Secure **Socket** (p. 1900) Layer for streaming based sockets.

`#include <src/main/decaf/net/ssl/SSLContext.h>`

Public Member Functions

- **SSLContext** (**SSLContextSpi** *contextImpl)
- virtual ~**SSLContext** ()
- **SocketFactory** * **getSocketFactory** ()

  *Returns an **SocketFactory** (p. 1916) instance for use with this Context, the **SocketFactory** (p. 1916) is owned by the Context and should not be deleted by the caller.*
- **ServerSocketFactory** * **getServerSocketFactory** ()

  *Returns an **ServerSocketFactory** (p. 1823) instance for use with this Context, the **ServerSocketFactory** (p. 1823) is owned by the Context and should not be deleted by the caller.*
- **SSLParameters** * **getDefaultSSLParameters** ()
- **SSLParameters** * **getSupportedSSLParameters** ()

Static Public Member Functions

- static **SSLContext** * **getDefault** ()

  *Gets the Default **SSLContext** (p. 1934).*
- static void **setDefault** (**SSLContext** *context)

  *Sets the default **SSLContext** (p. 1934) to be returned from future calls to getDefault.*

6.497.1 Detailed Description

Represents on implementation of the Secure **Socket** (p. 1900) Layer for streaming based sockets.

This class servers a a source of factories to be used to create new SSL **Socket** (p. 1900) instances.

Since

1.0

6.497.2 Constructor & Destructor Documentation

6.497.2.1 decaf::net::ssl::SSLContext::SSLContext (SSLContextSpi * *contextImpl*)

6.497.2.2 virtual decaf::net::ssl::SSLContext::~SSLContext ( ) [virtual]

6.497.3 Member Function Documentation

6.497.3.1 static SSLContext* decaf::net::ssl::SSLContext::getDefault ( ) [static]

Gets the Default **SSLContext** (p. 1934).

The default instance of the **SSLContext** (p. 1934) should be immediately usable without any need for the client to initialize this context.

Returns

a pointer to the Default **SSLContext** (p. 1934) instance.

6.497.3.2 SSLParameters* decaf::net::ssl::SSLContext::getDefaultSSLParameters ( )

Returns

a new instance of an **SSLParameters** (p. 1938) object containing the default set of settings for this **SSLContext** (p. 1934).

Exceptions

UnsupportedOperation-Exception	if the parameters cannot be retrieved.

6.497.3.3 ServerSocketFactory* decaf::net::ssl::SSLContext::getServerSocketFactory ( )

Returns an **ServerSocketFactory** (p. 1823) instance for use with this Context, the **ServerSocketFactory** (p. 1823) is owned by the Context and should not be deleted by the caller.

Returns

a pointer to this **SSLContext** (p. 1934)'s **ServerSocketFactory** (p. 1823) for creating **SSLServerSocket** (p. 1941) objects.

Exceptions

IllegalStateException	if the **SSLContextSpi** (p. 1936) requires initialization but it has not yet been initialized.

6.497.3.4 SocketFactory* decaf::net::ssl::SSLContext::getSocketFactory ( )

Returns an **SocketFactory** (p. 1916) instance for use with this Context, the **SocketFactory** (p. 1916) is owned by the Context and should not be deleted by the caller.

Returns

a pointer to this **SSLContext** (p. 1934)'s **SocketFactory** (p. 1916) for creating **SSLSocket** (p. 1947) objects.

Exceptions

IllegalStateException	if the **SSLContextSpi** (p. 1936) requires initialization but it has not yet been initialized.

6.497.3.5 SSLParameters* decaf::net::ssl::SSLContext::getSupportedSSLParameters ( )

Returns

a new instance of an **SSLParameters** (p. 1938) object containing the complete set of settings for this **SSLContext** (p. 1934).

Exceptions

UnsupportedOperationException	if the parameters cannot be retrieved.

6.497.3.6 static void decaf::net::ssl::SSLContext::setDefault (SSLContext * *context*) [static]

Sets the default **SSLContext** (p. 1934) to be returned from future calls to getDefault.

The set **SSLContext** (p. 1934) must be fully initialized and usable. The caller is responsible for deleting this object before the Library shutdown methods are called.

Exceptions

NullPointerException	if the context passed is NULL.

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLContext.h**

6.498 decaf::net::ssl::SSLContextSpi Class Reference

Defines the interface that should be provided by an **SSLContext** (p. 1934) provider.

`#include <src/main/decaf/net/ssl/SSLContextSpi.h>`

Inheritance diagram for decaf::net::ssl::SSLContextSpi:

Public Member Functions

- virtual ~**SSLContextSpi** ()
- virtual void **providerInit** (**security::SecureRandom** *random)=0

  Perform the initialization of this Context.
- virtual **SSLParameters** * **providerGetDefaultSSLParameters** ()

  *Creates a returns a new **SSLParameters** (p. 1938) instance that contains the default settings for this Providers **SSLContext** (p. 1934).*
- virtual **SSLParameters** * **providerGetSupportedSSLParameters** ()

  *Creates and returns a new **SSLParameters** (p. 1938) instance that contains the full set of supported parameters for this SSL Context.*
- virtual **SocketFactory** * **providerGetSocketFactory** ()=0

  *Returns a **SocketFactory** (p. 1916) instance that can be used to create new **SSLSocket** (p. 1947) objects.*

- virtual **ServerSocketFactory** * **providerGetServerSocketFactory** ()=0

  *Returns a **ServerSocketFactory** (p. 1823) instance that can be used to create new **SSLServerSocket** (p. 1941) objects.*

6.498.1 Detailed Description

Defines the interface that should be provided by an **SSLContext** (p. 1934) provider.

Since

1.0

6.498.2 Constructor & Destructor Documentation

6.498.2.1 virtual decaf::net::ssl::SSLContextSpi::~SSLContextSpi ( ) [virtual]

6.498.3 Member Function Documentation

6.498.3.1 virtual SSLParameters* decaf::net::ssl::SSLContextSpi::providerGetDefaultSSLParameters ( ) [virtual]

Creates a returns a new **SSLParameters** (p. 1938) instance that contains the default settings for this Providers **SSLContext** (p. 1934).

The returned **SSLParameters** (p. 1938) instance is requires to have non-empty values in its ciphersuites and protocols.

Returns

new **SSLParameters** (p. 1938) instance with the **SSLContext** (p. 1934) defaults.

Exceptions

UnsupportedOperationException	if the defaults cannot be obtained.

6.498.3.2 virtual ServerSocketFactory* decaf::net::ssl::SSLContextSpi::providerGetServerSocketFactory ( ) [pure virtual]

Returns a **ServerSocketFactory** (p. 1823) instance that can be used to create new **SSLServerSocket** (p. 1941) objects.

The **ServerSocketFactory** (p. 1823) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 1916) instance that can be used to create new SSLServerSockets.

Exceptions

IllegalStateException	if the **SSLContextSpi** (p. 1936) object requires initialization but has not been initialized yet.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 1549).

6.498.3.3 virtual SocketFactory* decaf::net::ssl::SSLContextSpi::providerGetSocketFactory ( ) [pure virtual]

Returns a **SocketFactory** (p. 1916) instance that can be used to create new **SSLSocket** (p. 1947) objects.

The **SocketFactory** (p. 1916) is owned by the Service Provider and should not be destroyed by the caller.

Returns

SocketFactory (p. 1916) instance that can be used to create new SSLSockets.

Exceptions

IllegalStateException	if the **SSLContextSpi** (p. 1936) object requires initialization but has not been initialized yet.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 1549).

6.498.3.4 virtual SSLParameters* decaf::net::ssl::SSLContextSpi::providerGetSupportedSSLParameters ( ) [virtual]

Creates and returns a new **SSLParameters** (p. 1938) instance that contains the full set of supported parameters for this SSL Context.

The returned **SSLParameters** (p. 1938) instance is requires to have non-empty values in its ciphersuites and protocols.

Returns

a new **SSLParameters** (p. 1938) instance with the full set of settings that are supported.

Exceptions

UnsupportedOperationException	if the supported parameters cannot be obtained.

6.498.3.5 virtual void decaf::net::ssl::SSLContextSpi::providerInit (security::SecureRandom * *random*) [pure virtual]

Perform the initialization of this Context.

Parameters

random	Pointer to an instance of a secure random number generator.

Exceptions

NullPointerException	if the SecureRandom instance is NULL.
KeyManagementException	if an error occurs while initializing the context.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 1550).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLContextSpi.h**

6.499 decaf::net::ssl::SSLParameters Class Reference

`#include <src/main/decaf/net/ssl/SSLParameters.h>`

Public Member Functions

- **SSLParameters** ()

  *Creates a new **SSLParameters** (p. 1938) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.*
- **SSLParameters** (**const** std::vector< std::string > &cipherSuites)

  *Creates a new **SSLParameters** (p. 1938) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.*
- **SSLParameters** (**const** std::vector< std::string > &cipherSuites, **const** std::vector< std::string > &protocols)

  *Creates a new **SSLParameters** (p. 1938) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.*
- virtual ~**SSLParameters** ()
- std::vector< std::string > **getCipherSuites** () **const**
- void **setCipherSuites** (**const** std::vector< std::string > &cipherSuites)

  Sets the vector of ciphersuites.
- std::vector< std::string > **getProtocols** () **const**
- void **setProtocols** (**const** std::vector< std::string > &protocols)

  Sets the vector of protocols.
- bool **getWantClientAuth** () **const**
- void **setWantClientAuth** (bool wantClientAuth)

  Sets whether client authentication should be requested.
- bool **getNeedClientAuth** () **const**
- void **setNeedClientAuth** (bool needClientAuth)

  Sets whether client authentication should be required.

6.499.1 Constructor & Destructor Documentation

6.499.1.1 decaf::net::ssl::SSLParameters::SSLParameters ( )

Creates a new **SSLParameters** (p. 1938) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.

6.499.1.2 decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & *cipherSuites*)

Creates a new **SSLParameters** (p. 1938) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.

Parameters

cipherSuites	The vector of cipherSuites for this **SSLParameters** (p. 1938) instance (can be empty).

6.499.1.3 decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & *cipherSuites,* const std::vector< std::string > & *protocols*)

Creates a new **SSLParameters** (p. 1938) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.

Parameters

cipherSuites	The vector of cipherSuites for this **SSLParameters** (p. 1938) instance (can be empty).
protocols	The vector of protocols for this **SSLParameters** (p. 1938) instance (can be empty).

6.499.1.4 virtual decaf::net::ssl::SSLParameters::~SSLParameters ( ) [virtual]

6.499.2 Member Function Documentation

6.499.2.1 std::vector<std::string> decaf::net::ssl::SSLParameters::getCipherSuites ( ) const [inline]

Returns

a copy of the vector of ciphersuites or an empty vector if none have been set.

6.499.2.2 bool decaf::net::ssl::SSLParameters::getNeedClientAuth ( ) const [inline]

Returns

whether client authentication should be required.

6.499.2.3 std::vector<std::string> decaf::net::ssl::SSLParameters::getProtocols ( ) const [inline]

Returns

a copy of the vector of protocols or an empty vector if none have been set.

6.499.2.4 bool decaf::net::ssl::SSLParameters::getWantClientAuth ( ) const [inline]

Returns

whether client authentication should be requested.

6.499.2.5 void decaf::net::ssl::SSLParameters::setCipherSuites (const std::vector< std::string > & *cipherSuites*) [inline]

Sets the vector of ciphersuites.

Parameters

cipherSuites	The vector of cipherSuites (can be an empty vector).

6.499.2.6 void decaf::net::ssl::SSLParameters::setNeedClientAuth (bool *needClientAuth*) [inline]

Sets whether client authentication should be required.

Calling this method clears the wantClientAuth flag.

Parameters

needClientAuth	whether client authentication should be required.

6.499.2.7 void decaf::net::ssl::SSLParameters::setProtocols (const std::vector< std::string > & *protocols*) [inline]

Sets the vector of protocols.

Parameters

protocols	the vector of protocols (or an empty vector)

6.499.2.8 void decaf::net::ssl::SSLParameters::setWantClientAuth (bool *wantClientAuth*) [inline]

Sets whether client authentication should be requested.

Calling this method clears the needClientAuth flag.

Parameters

whether	client authentication should be requested.

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLParameters.h**

6.500 decaf::net::ssl::SSLServerSocket Class Reference

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

`#include <src/main/decaf/net/ssl/SSLServerSocket.h>`

Inheritance diagram for decaf::net::ssl::SSLServerSocket:

Public Member Functions

- virtual ~**SSLServerSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () **const** =0

  *Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 1941).*
- virtual std::vector< std::string > **getSupportedProtocols** () **const** =0

  *Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 1941) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () **const** =0

  *Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 1941).*
- virtual void **setEnabledCipherSuites** (**const** std::vector< std::string > &suites)=0

  *Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 1941) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () **const** =0

  *Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 1941).*
- virtual void **setEnabledProtocols** (**const** std::vector< std::string > &protocols)=0

  *Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 1941) connection.*
- virtual bool **getWantClientAuth** () **const** =0
- virtual void **setWantClientAuth** (bool value)=0

  *Sets whether or not this **Socket** (p. 1900) will request Client Authentication.*
- virtual bool **getNeedClientAuth** () **const** =0
- virtual void **setNeedClientAuth** (bool value)=0

  *Sets whether or not this **Socket** (p. 1900) will require Client Authentication.*

Protected Member Functions

- **SSLServerSocket** ()

  Creates a non-bound server socket.
- **SSLServerSocket** (int port)

  *Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **SSLServerSocket** (int port, int backlog)

  *Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **SSLServerSocket** (int port, int backlog, **const decaf::net::InetAddress** *address)

  *Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.*

6.500.1 Detailed Description

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

The main function of this class is to create **SSLSocket** (p. 1947) objects by accepting connections from client sockets over SSL.

Since

1.0

6.500.2 Constructor & Destructor Documentation

6.500.2.1 decaf::net::ssl::SSLServerSocket::SSLServerSocket ( ) [protected]

Creates a non-bound server socket.

6.500.2.2 decaf::net::ssl::SSLServerSocket::SSLServerSocket (int *port*) [protected]

Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 1928) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 1921) is created.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.

Exceptions

IOException	if there is an I/O error while performing this operation.
IllegalArgumentException	if the port value is negative or greater than 65535.

6.500.2.3 decaf::net::ssl::SSLServerSocket::SSLServerSocket (int *port,* int *backlog*) [protected]

Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 1928) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 1921) is created.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The the number of incoming connection attempts to queue before connections are refused.

Exceptions

IOException	if there is an I/O error while performing this operation.
IllegalArgumentException	if the port value is negative or greater than 65535.

6.500.2.4 decaf::net::ssl::SSLServerSocket::SSLServerSocket (int *port,* int *backlog,* const decaf::net::InetAddress * *address*) [protected]

Creates a new **ServerSocket** (p. 1816) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 1928) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 1921) is created.

Parameters

port	The port to bind the **ServerSocket** (p. 1816) to.
backlog	The the number of incoming connection attempts to queue before connections are refused.
ifAddress	The IP Address to bind to on the local machine.

Exceptions

IOException	if there is an I/O error while performing this operation.
IllegalArgumentException	if the port value is negative or greater than 65535.

6.500.2.5 virtual decaf::net::ssl::SSLServerSocket::~SSLServerSocket ( ) [virtual]

6.500.3 Member Function Documentation

6.500.3.1 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledCipherSuites ( ) const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 1941).

Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1554).

6.500.3.2 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledProtocols ( ) const [pure virtual]

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 1941).

Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1554).

6.500.3.3 virtual bool decaf::net::ssl::SSLServerSocket::getNeedClientAuth ( ) const [pure virtual]

Returns

true if the **Socket** (p. 1900) requires client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1554).

6.500.3.4 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedCipherSuites ( ) const [pure virtual]

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 1941).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 1900).

Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1555).

6.500.3.5 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedProtocols ( ) const [pure virtual]

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 1941) instance.

Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1555).

6.500.3.6 virtual bool decaf::net::ssl::SSLServerSocket::getWantClientAuth ( ) const [pure virtual]

Returns

true if the **Socket** (p. 1900) request client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1555).

6.500.3.7 virtual void decaf::net::ssl::SSLServerSocket::setEnabledCipherSuites (const std::vector< std::string > & *suites*) [pure virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 1941) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites	An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1555).

6.500.3.8 virtual void decaf::net::ssl::SSLServerSocket::setEnabledProtocols (const std::vector< std::string > & *protocols*) [pure virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 1941) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols	An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1555).

6.500.3.9 virtual void decaf::net::ssl::SSLServerSocket::setNeedClientAuth (bool *value*) [pure virtual]

Sets whether or not this **Socket** (p. 1900) will require Client Authentication.

If set to true the **Socket** (p. 1900) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

Parameters

value	Whether the server socket should require client authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1556).

6.500.3.10 virtual void decaf::net::ssl::SSLServerSocket::setWantClientAuth (bool *value*) [pure virtual]

Sets whether or not this **Socket** (p. 1900) will request Client Authentication.

If set to true the **Socket** (p. 1900) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

Parameters

value	Whether the server socket should request client authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 1556).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLServerSocket.h**

6.501 decaf::net::ssl::SSLServerSocketFactory Class Reference

Factory class interface that provides methods to create SSL Server Sockets.

`#include <src/main/decaf/net/ssl/SSLServerSocketFactory.h>`

Inheritance diagram for decaf::net::ssl::SSLServerSocketFactory:

Public Member Functions

- virtual ∼**SSLServerSocketFactory** ()
- virtual std::vector< std::string > **getDefaultCipherSuites** ()=0

  Returns the list of cipher suites which are enabled by default.
- virtual std::vector< std::string > **getSupportedCipherSuites** ()=0

  Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Static Public Member Functions

- static **ServerSocketFactory** ∗ **getDefault** ()

  *Returns the current default SSL **ServerSocketFactory** (p. 1823), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

Protected Member Functions

- **SSLServerSocketFactory** ()

6.501.1 Detailed Description

Factory class interface that provides methods to create SSL Server Sockets.

Since

1.0

6.501.2 Constructor & Destructor Documentation

6.501.2.1 decaf::net::ssl::SSLServerSocketFactory::SSLServerSocketFactory ( ) [protected]

6.501.2.2 virtual decaf::net::ssl::SSLServerSocketFactory::∼SSLServerSocketFactory ( ) [virtual]

6.501.3 Member Function Documentation

6.501.3.1 static ServerSocketFactory* decaf::net::ssl::SSLServerSocketFactory::getDefault ( ) [static]

Returns the current default SSL **ServerSocketFactory** (p. 1823), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 1935)->getServerSocketFactory(). If that call fails, a non-functional factory is returned.

Returns

the default SSL **ServerSocketFactory** (p. 1823) pointer.

See also

decaf::net::ssl::SSLContext::getDefault() (p. 1935)

Reimplemented from **decaf::net::ServerSocketFactory** (p. 1825).

6.501.3.2 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getDefaultCipherSuites ( ) [pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 1947)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 1560), and **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 905).

6.501.3.3 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getSupportedCipherSuites ( ) [pure virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 1947)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 1560), and **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 906).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLServerSocketFactory.h**

6.502 decaf::net::ssl::SSLSocket Class Reference

`#include <src/main/decaf/net/ssl/SSLSocket.h>`

Inheritance diagram for decaf::net::ssl::SSLSocket:

Public Member Functions

- **SSLSocket** ()
- **SSLSocket** (**const InetAddress** *address, int port)

  *Creates a new **SSLSocket** (p. 1947) instance and connects it to the given address and port.*
- **SSLSocket** (**const InetAddress** *address, int port, **const InetAddress** *localAddress, int localPort)

  *Creates a new **SSLSocket** (p. 1947) instance and connects it to the given address and port.*
- **SSLSocket** (**const** std::string &host, int port)

  *Creates a new **SSLSocket** (p. 1947) instance and connects it to the given host and port.*
- **SSLSocket** (**const** std::string &host, int port, **const InetAddress** *localAddress, int localPort)

  *Creates a new **SSLSocket** (p. 1947) instance and connects it to the given host and port.*
- virtual ~**SSLSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () **const** =0

  *Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 1947).*
- virtual std::vector< std::string > **getSupportedProtocols** () **const** =0

  *Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 1947) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () **const** =0

  *Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 1900).*
- virtual void **setEnabledCipherSuites** (**const** std::vector< std::string > &suites)=0

  *Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 1900) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () **const** =0

  *Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 1900).*
- virtual void **setEnabledProtocols** (**const** std::vector< std::string > &protocols)=0

  *Sets the Protocols that are to be enabled on the SSL **Socket** (p. 1900) connection.*
- virtual **SSLParameters getSSLParameters** () **const**

  *Returns an **SSLParameters** (p. 1938) object for this **SSLSocket** (p. 1947) instance.*
- virtual void **setSSLParameters** (**const SSLParameters** &value)

  *Sets the **SSLParameters** (p. 1938) for this **SSLSocket** (p. 1947) using the supplied **SSLParameters** (p. 1938) instance.*
- virtual void **startHandshake** ()=0

  Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.
- virtual void **setUseClientMode** (bool value)=0

  Determines the mode that the socket uses when a handshake is initiated, client or server.
- virtual bool **getUseClientMode** () **const** =0

  *Gets whether this **Socket** (p. 1900) is in Client or Server mode, true indicates that the mode is set to Client.*
- virtual void **setNeedClientAuth** (bool value)=0

  *Sets the **Socket** (p. 1900) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getNeedClientAuth** () **const** =0

  Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.
- virtual void **setWantClientAuth** (bool value)=0

  *Sets the **Socket** (p. 1900) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

- virtual bool **getWantClientAuth** () **const** =0

  Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that cleints that do not send a certificate are not considered to have failed authentication.

6.502.1 Detailed Description

Since

1.0

6.502.2 Constructor & Destructor Documentation

6.502.2.1 decaf::net::ssl::SSLSocket::SSLSocket ( )

6.502.2.2 decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress * *address,* int *port*)

Creates a new **SSLSocket** (p. 1947) instance and connects it to the given address and port.

If the host parameter is empty then the loop back address is used.

Parameters

address	The address to connect to.
port	The port number to connect to [0...65535]

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
NullPointerException	if the **InetAddress** (p. 1113) instance in NULL.
IllegalArgumentException	if the port if not in range [0...65535]

6.502.2.3 decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress * *address,* int *port,* const InetAddress * *localAddress,* int *localPort*)

Creates a new **SSLSocket** (p. 1947) instance and connects it to the given address and port.

The **Socket** (p. 1900) will also bind to the local address and port specified.

Parameters

address	The address to connect to.
port	The port number to connect to [0...65535]
localAddress	The IP address on the local machine to bind to.
localPort	The port on the local machine to bind to.

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
NullPointerException	if the **InetAddress** (p. 1113) instance in NULL.
IllegalArgumentException	if the port if not in range [0...65535]

6.502.2.4 decaf::net::ssl::SSLSocket::SSLSocket (const std::string & *host,* int *port*)

Creates a new **SSLSocket** (p. 1947) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

Parameters

host	The host name or IP address to connect to, empty string means loopback.
port	The port number to connect to [0...65535]

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
IllegalArgumentException	if the port if not in range [0...65535]

6.502.2.5 decaf::net::ssl::SSLSocket::SSLSocket (const std::string & *host,* int *port,* const InetAddress * *localAddress,* int *localPort*)

Creates a new **SSLSocket** (p. 1947) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

Parameters

host	The host name or IP address to connect to, empty string means loopback.
port	The port number to connect to [0...65535]
localAddress	The IP address on the local machine to bind to.
localPort	The port on the local machine to bind to.

Exceptions

UnknownHostException *(p. 2181)*	if the host cannot be resolved.
IOException	if an I/O error occurs while connecting the **Socket** (p. 1900).
IllegalArgumentException	if the port if not in range [0...65535]

6.502.2.6 virtual decaf::net::ssl::SSLSocket::~SSLSocket ( ) [virtual]

6.502.3 Member Function Documentation

6.502.3.1 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledCipherSuites ( ) const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 1900).

Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1565).

6.502.3.2 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledProtocols ( ) const [pure virtual]

Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 1900).

Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1565).

6.502.3.3 virtual bool decaf::net::ssl::SSLSocket::getNeedClientAuth ( ) const [pure virtual]

Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1566).

6.502.3.4 virtual SSLParameters decaf::net::ssl::SSLSocket::getSSLParameters ( ) const [virtual]

Returns an **SSLParameters** (p. 1938) object for this **SSLSocket** (p. 1947) instance.

The cipherSuites and protocols vectors in the returned **SSLParameters** (p. 1938) reference will never be empty.

Returns

an **SSLParameters** (p. 1938) object with the settings in use for the **SSLSocket** (p. 1947).

6.502.3.5 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedCipherSuites ( ) const [pure virtual]

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 1947).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 1900).

Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1567).

6.502.3.6 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedProtocols ( ) const [pure virtual]

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 1947) instance.

Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1567).

6.502.3.7 virtual bool decaf::net::ssl::SSLSocket::getUseClientMode ( ) const [pure virtual]

Gets whether this **Socket** (p. 1900) is in Client or Server mode, true indicates that the mode is set to Client.

Returns

true if the **Socket** (p. 1900) is in Client mode, false otherwise.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1567).

6.502.3.8 virtual bool decaf::net::ssl::SSLSocket::getWantClientAuth ( ) const [pure virtual]

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that cleints that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1567).

6.502.3.9 virtual void decaf::net::ssl::SSLSocket::setEnabledCipherSuites (const std::vector< std::string > & *suites*) [pure virtual]

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 1900) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

Parameters

suites	An Vector of names for all the Cipher Suites that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1568).

6.502.3.10 virtual void decaf::net::ssl::SSLSocket::setEnabledProtocols (const std::vector< std::string > & *protocols*) [pure virtual]

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 1900) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

Parameters

protocols	An Vector of names for all the Protocols that are to be enabled.

Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1568).

6.502.3.11 virtual void decaf::net::ssl::SSLSocket::setNeedClientAuth (bool *value*) [pure virtual]

Sets the **Socket** (p. 1900) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

Parameters

value	The value indicating if a client is required to authenticate itself or not.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1569).

6.502.3.12 virtual void decaf::net::ssl::SSLSocket::setSSLParameters (const SSLParameters & *value*) [virtual]

Sets the **SSLParameters** (p. 1938) for this **SSLSocket** (p. 1947) using the supplied **SSLParameters** (p. 1938) instance.

If the cipherSutes vector in the **SSLParameters** (p. 1938) instance is not empty them the setEnabledCipherSuites method is called with that vector, if the protocols vector in the **SSLParameters** (p. 1938) instance is not empty then the setEnabledProtocols method is called with that vector. If the needClientAuth value or the wantClientAuth value is true then the setNeedClientAuth and setWantClientAuth methods are called respectively with a value of true, otherwise the setWantClientAuth method is called with a value of false.

Parameters

value	The **SSLParameters** (p. 1938) instance that is used to update this **SSLSocket** (p. 1947)'s settings.

Exceptions

IllegalArgumentException	if an error occurs while calling setEnabledCipherSuites or setEnabledProtocols.

6.502.3.13 virtual void decaf::net::ssl::SSLSocket::setUseClientMode (bool *value*) [pure virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 1900), once a handshake has be initiated this socket remains the the set mode; client or server, for the life of this object.

Parameters

value	The mode setting, true for client or false for server.

Exceptions

IllegalArguementException	if the handshake process has begun and mode is lcoked.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1569).

6.502.3.14 virtual void decaf::net::ssl::SSLSocket::setWantClientAuth (bool *value*) [pure virtual]

Sets the **Socket** (p. 1900) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

Parameters

value	The value indicating if a client is requested to authenticate itself or not.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1570).

6.502.3.15 virtual void decaf::net::ssl::SSLSocket::startHandshake ( ) [pure virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not require to support multiple handshakes and can throw an IOException to indicate an error.

Exceptions

IOException	if an I/O error occurs while performing the Handshake

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 1570).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLSocket.h**

6.503 decaf::net::ssl::SSLSocketFactory Class Reference

Factory class interface for a **SocketFactory** (p. 1916) that can create **SSLSocket** (p. 1947) objects.

`#include <src/main/decaf/net/ssl/SSLSocketFactory.h>`

Inheritance diagram for decaf::net::ssl::SSLSocketFactory:

Public Member Functions

- virtual ~**SSLSocketFactory** ()
- virtual std::vector< std::string > **getDefaultCipherSuites** ()=0

  Returns the list of cipher suites which are enabled by default.
- virtual std::vector< std::string > **getSupportedCipherSuites** ()=0

  Returns the names of the cipher suites which could be enabled for use on an SSL connection.
- virtual **Socket** * **createSocket** (**Socket** *socket, std::string host, int port, bool autoClose)=0

  Returns a socket layered over an existing socket connected to the named host, at the given port.

Static Public Member Functions

- static **SocketFactory** * **getDefault** ()

  *Returns the current default SSL **SocketFactory** (p. 1916), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

Protected Member Functions

- **SSLSocketFactory** ()

6.503.1 Detailed Description

Factory class interface for a **SocketFactory** (p. 1916) that can create **SSLSocket** (p. 1947) objects.

Since

1.0

6.503.2 Constructor & Destructor Documentation

6.503.2.1 decaf::net::ssl::SSLSocketFactory::SSLSocketFactory ( ) [protected]

6.503.2.2 virtual decaf::net::ssl::SSLSocketFactory::~SSLSocketFactory ( ) [virtual]

6.503.3 Member Function Documentation

6.503.3.1 virtual Socket* decaf::net::ssl::SSLSocketFactory::createSocket (Socket * *socket,* std::string *host,* int *port,* bool *autoClose*) [pure virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

Parameters

socket	The existing socket to layer over.
host	The server host the original **Socket** (p. 1900) is connected to.
port	The server port the original **Socket** (p. 1900) is connected to.
autoClose	Should the layered over **Socket** (p. 1900) be closed when the topmost socket is closed.

Returns

a new **Socket** (p. 1900) instance that wraps the given **Socket** (p. 1900).

Exceptions

IOException	if an I/O exception occurs while performing this operation.
UnknownHostException *(p. 2181)*	if the host is unknown.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1579), and **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 911).

6.503.3.2 static SocketFactory* decaf::net::ssl::SSLSocketFactory::getDefault ( ) [static]

Returns the current default SSL **SocketFactory** (p. 1916), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 1935)->getSocketFactory(). If that call fails, a non-functional factory is returned.

Returns

the default SSL **SocketFactory** (p. 1916) pointer.

See also

decaf::net::ssl::SSLContext::getDefault() (p. 1935)

Reimplemented from **decaf::net::SocketFactory** (p. 1920).

6.503.3.3 virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getDefaultCipherSuites ( ) [pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

Returns

an STL vector containing the list of cipher suites enabled by default.

See also

getSupportedCipherSuites() (p. 1956)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1579), and **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 911).

6.503.3.4 virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getSupportedCipherSuites ( ) [pure virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns

an STL vector containing the list of supported cipher suites.

See also

getDefaultCipherSuites() (p. 1956)

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 1580), and **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 912).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLSocketFactory.h**

6.504 activemq::transport::tcp::SslTransport Class Reference

Transport (p. 2161) for connecting to a Broker using an SSL Socket.

`#include <src/main/activemq/transport/tcp/SslTransport.h>`

Inheritance diagram for activemq::transport::tcp::SslTransport:

Public Member Functions

- **SslTransport** (**const Pointer**< **Transport** > &**next**)

  *Creates a new instance of the **SslTransport** (p. 1956), the transport will not attempt to connect to a remote host until the connect method is called.*
- virtual ~**SslTransport** ()

Protected Member Functions

- virtual **decaf::net::Socket** * **createSocket** ()

  Create an unconnected Socket instance to be used by the transport to communicate with the broker.

  Returns

  a newly created unconnected Socket instance.

  Exceptions

IOException	*if there is an error while creating the unconnected Socket.*

- virtual void **configureSocket** (**decaf::net::Socket** *socket, **decaf::util::Properties** &properties)

6.504.1 Detailed Description

Transport (p. 2161) for connecting to a Broker using an SSL Socket.

This transport simply wraps the **TcpTransport** (p. 2086) and provides the **TcpTransport** (p. 2086) an SSL based Socket pointer allowing the core **TcpTransport** (p. 2086) logic to be reused.

Since

3.2.0

6.504.2 Constructor & Destructor Documentation

6.504.2.1 activemq::transport::tcp::SslTransport::SslTransport (const Pointer< Transport > & *next*)

Creates a new instance of the **SslTransport** (p. 1956), the transport will not attempt to connect to a remote host until the connect method is called.

Parameters

next	the next transport in the chain

6.504.2.2 virtual activemq::transport::tcp::SslTransport::~SslTransport ( ) [virtual]

6.504.3 Member Function Documentation

6.504.3.1 virtual void activemq::transport::tcp::SslTransport::configureSocket (decaf::net::Socket * *socket,* decaf::util::Properties & *properties*) [protected, virtual]

6.504.3.2 virtual decaf::net::Socket* activemq::transport::tcp::SslTransport::createSocket ( ) [protected, virtual]

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

Returns

a newly created unconnected Socket instance.

Exceptions

IOException	if there is an error while creating the unconnected Socket.

Reimplemented from **activemq::transport::tcp::TcpTransport** (p. 2088).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**SslTransport.h**

6.505 activemq::transport::tcp::SslTransportFactory Class Reference

`#include <src/main/activemq/transport/tcp/SslTransportFactory.h>`

Inheritance diagram for activemq::transport::tcp::SslTransportFactory:

Public Member Functions

- virtual ~**SslTransportFactory** ()

Protected Member Functions

- virtual **Pointer**< **Transport** > **doCreateComposite** (**const decaf::net::URI** &location, **const Pointer**< **wireformat::WireFormat** > &wireFormat, **const decaf::util::Properties** &properties)

6.505.1 Constructor & Destructor Documentation

6.505.1.1 virtual activemq::transport::tcp::SslTransportFactory::~SslTransportFactory ( ) [virtual]

6.505.2 Member Function Documentation

6.505.2.1 virtual Pointer<Transport> activemq::transport::tcp::SslTransportFactory::doCreateComposite (const decaf::net::URI & *location,* const Pointer< wireformat::WireFormat > & *wireFormat,* const decaf::util::Properties & *properties*) [protected, virtual]

Reimplemented from **activemq::transport::tcp::TcpTransportFactory** (p. 2090).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**SslTransportFactory.h**

6.506 activemq::commands::BrokerError::StackTraceElement Struct Reference

`#include <src/main/activemq/commands/BrokerError.h>`

Public Member Functions

- **StackTraceElement** ()

Data Fields

- std::string **ClassName**
- std::string **FileName**
- std::string **MethodName**
- int **LineNumber**

6.506.1 Constructor & Destructor Documentation

6.506.1.1 activemq::commands::BrokerError::StackTraceElement::StackTraceElement ( ) [inline]

6.506.2 Field Documentation

6.506.2.1 std::string activemq::commands::BrokerError::StackTraceElement::ClassName

6.506.2.2 std::string activemq::commands::BrokerError::StackTraceElement::FileName

6.506.2.3 int activemq::commands::BrokerError::StackTraceElement::LineNumber

6.506.2.4 std::string activemq::commands::BrokerError::StackTraceElement::MethodName

The documentation for this struct was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

6.507 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** ()
- virtual ~**StandardErrorOutputStream** ()
- virtual void **flush** ()

  Flushes this stream by writing any buffered output to the underlying stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

The default implementation of this method does nothing.

- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
  The object is generally no longer usable after calling close.

  Exceptions

IOException (p. 1198)	*if an error occurs while closing.*

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (**const** unsigned char *buffer, int size, int offset, int length)

6.507.1 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`.

6.507.2 Constructor & Destructor Documentation

6.507.2.1 decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ( )

6.507.2.2 virtual decaf::internal::io::StandardErrorOutputStream::~StandardErrorOutputStream ( ) [virtual]

6.507.3 Member Function Documentation

6.507.3.1 virtual void decaf::internal::io::StandardErrorOutputStream::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(p. 1198)*	if an error occurs while closing.

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.507.3.2 virtual void decaf::internal::io::StandardErrorOutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.507.3.3 virtual void decaf::internal::io::StandardErrorOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 1602).

6.507.3.4 virtual void decaf::internal::io::StandardErrorOutputStream::flush ( ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardErrorOutputStream.h**

6.508 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 int **available** () **const**

  Indicates the number of bytes available.

Protected Member Functions

- virtual int **doReadByte** ()

6.508.1 Constructor & Destructor Documentation

6.508.1.1 decaf::internal::io::StandardInputStream::StandardInputStream ( )

6.508.1.2 virtual decaf::internal::io::StandardInputStream::~StandardInputStream ( ) [virtual]

6.508.2 Member Function Documentation

6.508.2.1 virtual int decaf::internal::io::StandardInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.508.2.2 virtual int decaf::internal::io::StandardInputStream::doReadByte ( ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1137).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardInputStream.h**

6.509 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 **flush** ()

  Flushes this stream by writing any buffered output to the underlying stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

  The default implementation of this method does nothing.
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
  The object is generally no longer usable after calling close.

  Exceptions

IOException (p. 1198)	*if an error occurs while closing.*

  The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)
- virtual void **doWriteArrayBounded** (**const** unsigned char ∗buffer, int size, int offset, int length)

6.509.1 Constructor & Destructor Documentation

6.509.1.1 decaf::internal::io::StandardOutputStream::StandardOutputStream ( )

6.509.1.2 virtual decaf::internal::io::StandardOutputStream::~StandardOutputStream ( ) [virtual]

6.509.2 Member Function Documentation

6.509.2.1 virtual void decaf::internal::io::StandardOutputStream::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(*p. *1198)*	if an error occurs while closing.

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.509.2.2 virtual void decaf::internal::io::StandardOutputStream::doWriteArrayBounded (const unsigned char * *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.509.2.3 virtual void decaf::internal::io::StandardOutputStream::doWriteByte (unsigned char *value*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 1602).

6.509.2.4 virtual void decaf::internal::io::StandardOutputStream::flush ( ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs.

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

6.510 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** () throw ()
- virtual void **start** ()=0

  Starts the service.

6.510.1 Detailed Description

Interface for a class that implements the start method.

An object that implements the **Startable** (p. 1963) 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

6.510.2 Constructor & Destructor Documentation

6.510.2.1 virtual cms::Startable::~Startable ( ) throw () [virtual]

6.510.3 Member Function Documentation

6.510.3.1 virtual void cms::Startable::start ( ) [pure virtual]

Starts the service.

Exceptions

CMSException *(p. 640)*	if an internal error occurs while starting.

Implemented in **activemq::core::ActiveMQConnection** (p. 164), **activemq::core::ActiveMQSession** (p. 273), **activemq::core::ActiveMQConsumer** (p. 189), **activemq::cmsutil::PooledSession** (p. 1632), and **activemq::cmsutil::CachedConsumer** (p. 572).

Referenced by activemq::cmsutil::PooledSession::start().

The documentation for this class was generated from the following file:

- src/main/cms/**Startable.h**

6.511 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**

6.512 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**

6.512.1 Constructor & Destructor Documentation

6.512.1.1 activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer ( )

6.512.1.2 virtual activemq::core::ActiveMQConstants::StaticInitializer::~StaticInitializer ( ) [inline, virtual]

6.512.2 Field Documentation

6.512.2.1 std::map<std::string, DestinationOption> activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap [static]

6.512.2.2 std::string activemq::core::ActiveMQConstants::StaticInitializer::destOptions[NUM_OPTIONS] [static]

6.512.2.3 std::string activemq::core::ActiveMQConstants::StaticInitializer::uriParams[NUM_PARAMS] [static]

6.512.2.4 std::map<std::string, URIParam> activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap [static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConstants.h**

6.513 decaf::util::StlList< E > Class Template Reference

List (p. 1286) 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 bool **equals** (**const Collection**< E > &collection) **const**

  *Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.*

  Parameters

collection	*- The **Collection** (p. 660) to be compared to this one.*

  Returns

  *true if this **Collection** (p. 660) is equal to the one given.*

- virtual void **copy** (**const Collection**< E > &collection)

  *Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).*
  *The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 660)'s clear method.*

  Parameters

collection	*The collection to mirror.*

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
IllegalStateException	*if the elements cannot be added at this time due to insertion restrictions.*

- virtual **Iterator**< E > * **iterator** ()

  Returns

  an iterator over a set of elements of type T.

- 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** (int index)

  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. 669))*

- virtual **ListIterator**< E > * **listIterator** (int index) **const**
- virtual void **clear** ()

  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. 1210) 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

UnsupportedOperation-Exception	*if the clear operation is not supported by this collection*

- virtual bool **isEmpty** () **const**

  Returns true if this collection contains no elements.
  *This implementation returns **size()** (p. 669) == 0.*

  Returns

  true if the size method return 0.

- virtual int **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** (int index) **const**

  Gets the element contained at position passed.

  Parameters

index	*The position to get.*

  Returns

  value at index specified.

  Exceptions

IndexOutOfBoundsException	*if the index given is less than zero or greater than the **List** (p. 1286) size.*

- virtual E **set** (int index, **const** E &element)

  Replaces the element at the specified position in this list with the specified element.

  Parameters

index	*The index of the element to replace.*
element	*The element to be stored at the specified position.*

  Returns

  the element previously at the specified position.

  Exceptions

IndexOutOfBoundsException	*if the index given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

- virtual void **add** (int index, **const** E &element)

  Inserts the specified element at the specified position in this list.

- virtual bool **add** (**const** E &value)

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

- virtual bool **addAll** (**const Collection**< E > &collection)

  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	*The **Collection** (p. 660) whose elements are added to this one.*

  Returns

  true if this collection changed as a result of the call

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
IllegalArgumentException	*if some property of an element prevents it from being added to this collection*
IllegalStateException	*if an element cannot be added at this time due to insertion restrictions.*

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).

- virtual bool **addAll** (int index, **const Collection**< E > &collection)

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. 660) 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 given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

- virtual bool **remove** (**const** E &value)

Removes a single instance of the specified element from the collection.
More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this **Collection** (p. 660).*

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*

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.

- virtual E **removeAt** (int index)

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 given is less than zero or greater than the **List** (p. 1286) size.*
UnsupportedOperationExceptio	*if this is an unmodifiable collection.*

- virtual int **indexOf** (**const** E &value) **const**

  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 int **lastIndexOf** (**const** E &value) **const**

  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 bool **contains** (**const** E &value) **const**

  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 (value == NULL ? e == NULL : value == e).

  Parameters

value	*The value to check for presence in the collection.*

  Returns

  true if there is at least one of the elements in the collection

  Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

  This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

6.513.1 Detailed Description

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

List (p. 1286) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

6.513.2 Constructor & Destructor Documentation

6.513.2.1 template<typename E> decaf::util::StlList< E >::StlList ( ) [inline]

Default constructor - does nothing.

6.513.2.2 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.

References decaf::util::StlList< E >::copy().

6.513.2.3 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.

References decaf::util::StlList< E >::copy().

6.513.3 Member Function Documentation

6.513.3.1 template<typename E> virtual void decaf::util::StlList< E >::add (int *index*, const E & *element*) [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	The index at which the specified element is to be inserted.
element	The element to be inserted in this **List** (p. 1286).

Exceptions

IndexOutOfBounds-Exception	if the index is greater than size of the **List** (p. 1286).
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List< E >** (p. 1287).

References decaf::util::StlList< E >::size().

6.513.3.2 template<typename E> virtual bool decaf::util::StlList< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Reimplemented from **decaf::util::AbstractList< E >** (p. 98).

6.513.3.3 template<typename E> virtual bool decaf::util::StlList< E >::addAll (const Collection< E > & *collection*) [inline, 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	The **Collection** (p. 660) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
IllegalArgumentException	if some property of an element prevents it from being added to this collection
IllegalStateException	if an element cannot be added at this time due to insertion restrictions.

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).

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).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 87).

References decaf::util::Collection< E >::isEmpty(), decaf::util::StlList< E >::listIterator(), and decaf::util::-Collection< E >::toArray().

6.513.3.4 template<typename E> virtual bool decaf::util::StlList< E >::addAll (int *index*, const Collection< E > & *collection*) [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. 660) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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.

Reimplemented from **decaf::util::AbstractList< E >** (p. 99).

References decaf::util::Collection< E >::isEmpty(), decaf::util::StlList< E >::listIterator(), decaf::util::StlList< E >::size(), and decaf::util::Collection< E >::toArray().

6.513.3.5 template<typename E> virtual void decaf::util::StlList< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractList< E >** (p. 100).

6.513.3.6 template<typename E> virtual bool decaf::util::StlList< E >::contains (const E & *value*) const [inline, 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 (value == NULL ?

e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 88).

6.513.3.7 template<typename E> virtual void decaf::util::StlList< E >::copy (const Collection< E > & *collection*) [inline, virtual]

Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 660)'s clear method.

Parameters

collection	The collection to mirror.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
IllegalStateException	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 89).

Referenced by decaf::util::StlList< E >::StlList().

6.513.3.8 template<typename E> virtual bool decaf::util::StlList< E >::equals (const Collection< E > & *collection*) const [inline, virtual]

Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.

Parameters

collection	- The **Collection** (p. 660) to be compared to this one.

Returns

true if this **Collection** (p. 660) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

6.513.3.9 template<typename E> virtual E decaf::util::StlList< E >::get (int *index*) const [inline, virtual]

Gets the element contained at position passed.

Parameters

index	The position to get.

Returns

value at index specified.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.

Implements **decaf::util::List< E >** (p. 1289).

References decaf::util::StlList< E >::size().

6.513.3.10 template<typename E> virtual int decaf::util::StlList< E >::indexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Reimplemented from **decaf::util::AbstractList< E >** (p. 100).

6.513.3.11 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. 669) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

6.513.3.12 template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Reimplemented from **decaf::util::AbstractList< E >** (p. 101).

6.513.3.13 template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator ( ) const [inline, virtual]

Reimplemented from **decaf::util::AbstractList< E >** (p. 101).

6.513.3.14 template<typename E> virtual int decaf::util::StlList< E >::lastIndexOf (const E & *value*) const [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	The element to search for in this **List** (p. 1286).

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

Reimplemented from **decaf::util::AbstractList< E >** (p. 102).

References decaf::util::StlList< E >::size().

6.513.3.15 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).

Reimplemented from **decaf::util::AbstractList< E >** (p. 102).

Referenced by decaf::util::StlList< E >::addAll().

6.513.3.16 template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator ( ) const [inline, virtual]

Reimplemented from **decaf::util::AbstractList< E >** (p. 103).

6.513.3.17 template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (int index) [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

IndexOutOfBounds-Exception	if the index is out of range (index < 0 || index > **size()** (p. 669))

Reimplemented from **decaf::util::AbstractList< E >** (p. 103).

References decaf::util::StlList< E >::size().

6.513.3.18 template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (int index) const [inline, virtual]

Reimplemented from **decaf::util::AbstractList< E >** (p. 103).

References decaf::util::StlList< E >::size().

6.513.3.19 template<typename E> virtual bool decaf::util::StlList< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 92).

References decaf::util::StlList< E >::size().

6.513.3.20 template<typename E> virtual E decaf::util::StlList< E >::removeAt (int index) [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

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.

Reimplemented from **decaf::util::AbstractList< E >** (p. 104).

References decaf::util::StlList< E >::size().

6.513.3.21 template<typename E> virtual E decaf::util::StlList< E >::set (int index , const E & *element*) [inline, virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters

index	The index of the element to replace.
element	The element to be stored at the specified position.

Returns

the element previously at the specified position.

Exceptions

IndexOutOfBounds-Exception	if the index given is less than zero or greater than the **List** (p. 1286) size.
UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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::List< E >** (p. 1294).

References decaf::util::StlList< E >::size().

6.513.3.22 template<typename E> virtual int 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. 669).

Referenced by decaf::util::StlList< E >::add(), decaf::util::StlList< E >::addAll(), decaf::util::StlList< E >::get(), decaf::util::StlList< E >::lastIndexOf(), decaf::util::StlList< E >::listIterator(), decaf::util::StlList< E >::remove(), decaf::util::StlList< E >::removeAt(), and decaf::util::StlList< E >::set().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlList.h**

6.514 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 1371) 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. 1371) to compare to this one.*

  Returns

  *true if the **Map** (p. 1371) 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** ()

  Removes all keys and values from this map.

  Exceptions

UnsupportedOperation-Exception	*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. 1371) contains any element or not, TRUE or FALSE*

- virtual int **size** () **const**

  Returns

  The number of elements (key/value pairs) in this map.

- virtual V & **get** (**const** K &key)

  *Gets the value mapped to the specified key in the **Map** (p. 1371).*
  *If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.*

  Parameters

key	*The search key.*

  Returns

  A reference to the value for the given key.

  Exceptions

NoSuchElementException (p. 1537)	*if the key requests doesn't exist in the **Map** (p. 1371).*

- virtual **const** V & **get** (**const** K &key) **const**

  *Gets the value mapped to the specified key in the **Map** (p. 1371).*
  *If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.*

  Parameters

key	*The search key.*

  Returns

  A {const} reference to the value for the given key.

Exceptions

NoSuchElementException (p. 1537)	*if the key requests doesn't exist in the **Map** (p. 1371).*

- virtual void **put** (**const** K &key, **const** V &value)

  Sets the value for the specified key.

  Parameters

key	*The target key.*
value	*The value to be set.*

  Exceptions

UnsupportedOperation-Exception	*if this map is unmodifiable.*

- virtual void **putAll** (**const StlMap**< K, V, COMPARATOR > &other)

- virtual void **putAll** (**const Map**< K, V, COMPARATOR > &other)

  *Stores a copy of the Mappings contained in the other **Map** (p. 1371) in this one.*

  Parameters

other	*A **Map** (p. 1371) instance whose elements are to all be inserted in this **Map** (p. 1371).*

  Exceptions

UnsupportedOperation-Exception	*If the implementing class does not support the putAll operation.*

- virtual V **remove** (**const** K &key)

  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 (p. 1537)	*if this key is not in the **Map** (p. 1371).*
UnsupportedOperation-Exception	*if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () **const**

  *Returns a **Set** (p. 1857) 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. 1210), **Set.remove** (p. 667), 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** ()

  Locks the object.

- virtual bool **tryLock** ()

  Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

Unlocks the object.

- virtual void **wait** ()

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

- virtual void **wait** (long long millisecs)

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

- virtual void **wait** (long long millisecs, int nanos)

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

- virtual void **notify** ()

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

- virtual void **notifyAll** ()

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

6.514.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>>class decaf::util::StlMap< K, V, COMPARATOR >

Map (p. 1371) 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

6.514.2 Constructor & Destructor Documentation

6.514.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap ( ) [inline]

Default constructor - does nothing.

6.514.2.2 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.

6.514.2.3 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.

6.514.2.4 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::StlMap< K, V, COMPARATOR >::~StlMap ( ) [inline, virtual]

6.514.3 Member Function Documentation

6.514.3.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::clear ( ) [inline, virtual]

Removes all keys and values from this map.

Exceptions

UnsupportedOperation-Exception	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1373).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::copy().

6.514.3.2 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. 1373).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals().

6.514.3.3 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. 1374).

6.514.3.4 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().

6.514.3.5 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. 1374).

6.514.3.6 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]

6.514.3.7 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. 1371) to compare to this one.

Returns

true if the **Map** (p. 1371) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1374).

6.514.3.8 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::StlMap< K, V, COMPARATOR >::get (const K & *key*) [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1371).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.

Parameters

key	The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException *(p. 1537)*	if the key requests doesn't exist in the **Map** (p. 1371).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1375).

6.514.3.9 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::StlMap< K, V, COMPARATOR >::get (const K & *key*) const [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1371).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** (p. 1537) is thrown.

Parameters

key	The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException *(p. 1537)*	if the key requests doesn't exist in the **Map** (p. 1371).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1375).

6.514.3.10 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. 1371) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1376).

6.514.3.11 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. 1857) 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. 1210), **Set.remove** (p. 667), 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. 1376).

6.514.3.12 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::lock ( ) [inline, virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

6.514.3.13 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notify ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2047).

6.514.3.14 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notifyAll ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2048).

6.514.3.15 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*) [inline, virtual]

Sets the value for the specified key.

Parameters

key	The target key.
value	The value to be set.

Exceptions

UnsupportedOperation-Exception	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1377).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::putAll().

6.514.3.16 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*) [inline, virtual]

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::copy().

6.514.3.17 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*) [inline, virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 1371) in this one.

Parameters

other	A **Map** (p. 1371) instance whose elements are to all be inserted in this **Map** (p. 1371).

Exceptions

UnsupportedOperation-Exception	If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1377).

6.514.3.18 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove (const K & *key*) [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 *(p. 1537)*	if this key is not in the **Map** (p. 1371).
UnsupportedOperation-Exception	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1378).

6.514.3.19 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual int 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. 1378).

6.514.3.20 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::tryLock ( ) [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. 2048).

6.514.3.21 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::unlock ( ) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

6.514.3.22 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. 1379).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::values().

6.514.3.23 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.514.3.24 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.514.3.25 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlMap.h**

6.515 decaf::util::StlQueue< T > Class Template Reference

The **Queue** (p. 1723) 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. 1209) over this **Queue** (p. 1723).*
- 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. 1723).*
- bool **empty** () **const**

  *Checks if this **Queue** (p. 1723) 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** ()

  Locks the object.
- virtual bool **tryLock** ()

  Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** ()

  Unlocks the object.
- virtual void **wait** ()

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()

  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()

  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.

6.515.1 Detailed Description

template<typename T>class decaf::util::StlQueue< T >

The **Queue** (p. 1723) class accepts messages with an psuh(m) command where m is the message to be queued.

It destructively returns the message with **pop()** (p. 1992). **pop()** (p. 1992) returns messages in the order they were enqueued.

Queue (p. 1723) 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. 1723) 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. 1723).

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.

6.515.2 Constructor & Destructor Documentation

6.515.2.1 template<typename T> decaf::util::StlQueue< T >::StlQueue ( ) [inline]

6.515.2.2 template<typename T> virtual decaf::util::StlQueue< T >::~StlQueue ( ) [inline, virtual]

6.515.3 Member Function Documentation

6.515.3.1 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)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.2 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)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.3 template<typename T> void decaf::util::StlQueue< T >::clear ( ) [inline]

Empties this queue.

6.515.3.4 template<typename T> bool decaf::util::StlQueue< T >::empty ( ) const [inline]

Checks if this **Queue** (p. 1723) is currently empty.

Returns

boolean indicating queue emptiness

6.515.3.5 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. 1723) Object Type reference.

6.515.3.6 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)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.7 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)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.8 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< T >::back(), decaf::util::StlQueue< T >::front(), and decaf::util::StlQueue< T >::pop().

6.515.3.9 template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator ( ) [inline]

Gets an **Iterator** (p. 1209) over this **Queue** (p. 1723).

Returns

new iterator pointer that is owned by the caller.

6.515.3.10 template<typename T> virtual void decaf::util::StlQueue< T >::lock ( ) [inline, virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

References decaf::util::concurrent::Mutex::lock().

6.515.3.11 template<typename T> virtual void decaf::util::StlQueue< T >::notify ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2047).

References decaf::util::concurrent::Mutex::notify().

6.515.3.12 template<typename T> virtual void decaf::util::StlQueue< T >::notifyAll ( ) [inline, virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2048).

References decaf::util::concurrent::Mutex::notifyAll().

6.515.3.13 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)

References decaf::util::StlQueue< T >::getSafeValue().

6.515.3.14 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. 1723) Object Type reference.

6.515.3.15 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.

6.515.3.16 template<typename T> size_t decaf::util::StlQueue< T >::size ( ) const [inline]

Gets the Number of elements currently in the **Queue** (p. 1723).

Returns

Queue (p. 1723) Size

6.515.3.17 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.

6.515.3.18 template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock ( ) [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. 2048).

References decaf::util::concurrent::Mutex::tryLock().

6.515.3.19 template<typename T> virtual void decaf::util::StlQueue< T >::unlock ( ) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

References decaf::util::concurrent::Mutex::unlock().

6.515.3.20 template<typename T> virtual void decaf::util::StlQueue< T >::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

References decaf::util::concurrent::Mutex::wait().

6.515.3.21 template<typename T> virtual void decaf::util::StlQueue< T >::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

References decaf::util::concurrent::Mutex::wait().

6.515.3.22 template<typename T> virtual void decaf::util::StlQueue< T >::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

References decaf::util::concurrent::Mutex::wait().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlQueue.h**

6.516 decaf::util::StlSet< E > Class Template Reference

Set (p. 1857) 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** ()

  Returns

  an iterator over a set of elements of type T.
- **Iterator**< E > * **iterator** () **const**
- virtual bool **equals** (**const Collection**< E > &collection) **const**

  *Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.*

  Parameters

collection	- *The **Collection** (p. 660) to be compared to this one.*

  Returns

  *true if this **Collection** (p. 660) is equal to the one given.*
- virtual void **copy** (**const Collection**< E > &collection)

  *Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).*

*The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 660)'s clear method.*

Parameters

collection	*The collection to mirror.*

Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
IllegalStateException	*if the elements cannot be added at this time due to insertion restrictions.*

- virtual void **clear** ()

  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. 1210) 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

UnsupportedOperation-Exception	*if the clear operation is not supported by this collection*

- virtual bool **contains** (**const** E &value) **const**

  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 (value == NULL ? e == NULL : value == e).

  Parameters

value	*The value to check for presence in the collection.*

  Returns

  true if there is at least one of the elements in the collection

  Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

  This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

- virtual bool **isEmpty** () **const**
- virtual int **size** () **const**
- virtual bool **add** (**const** E &value)

  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. 660) 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	*The reference to the element to add to this **Collection** (p. 660).*

  Returns

  *true if the element was added to this **Collection** (p. 660).*

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.*
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.*

- virtual bool **remove** (**const** E &value)

  Removes a single instance of the specified element from the collection.
  More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	*The reference to the element to remove from this* ***Collection*** *(*p. *660).*

  Returns

  true if the collection was changed, false otherwise.

  Exceptions

UnsupportedOperationExceptio	*if this is an unmodifiable collection.*
NullPointerException	*if the* ***Collection*** *(*p. *660) is a container of pointers and does not allow NULL values.*

  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.

6.516.1 Detailed Description

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

Set (p. 1857) 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.

6.516.2 Constructor & Destructor Documentation

6.516.2.1 template<typename E> decaf::util::StlSet< E >::StlSet ( ) [inline]

Default constructor - does nothing.

6.516.2.2 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.

6.516.2.3 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.

6.516.2.4 template<typename E> virtual decaf::util::StlSet< E >::~StlSet ( ) [inline, virtual]

6.516.3 Member Function Documentation

6.516.3.1 template<typename E> virtual bool decaf::util::StlSet< E >::add (const E & *value*) [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. 660) 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	The reference to the element to add to this **Collection** (p. 660).

Returns

true if the element was added to this **Collection** (p. 660).

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.
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. 662).

6.516.3.2 template<typename E> virtual void decaf::util::StlSet< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 88).

6.516.3.3 template<typename E> virtual bool decaf::util::StlSet< E >::contains (const E & *value*) const [inline, 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 (value == NULL ? e == NULL : value == e).

Parameters

value	The value to check for presence in the collection.

Returns

true if there is at least one of the elements in the collection

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 88).

6.516.3.4 template<typename E> virtual void decaf::util::StlSet< E >::copy (const Collection< E > & *collection*) [inline, virtual]

Renders this **Collection** (p. 660) as a Copy of the given **Collection** (p. 660).

The default implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection** (p. 660)'s clear method.

Parameters

collection	The collection to mirror.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
IllegalStateException	if the elements cannot be added at this time due to insertion restrictions.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 89).

Referenced by decaf::util::StlSet< Resource * >::copy(), and decaf::util::StlSet< Resource * >::StlSet().

6.516.3.5 template<typename E> virtual bool decaf::util::StlSet< E >::equals (const Collection< E > & *collection*) const [inline, virtual]

Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.

Parameters

collection	- The **Collection** (p. 660) to be compared to this one.

Returns

true if this **Collection** (p. 660) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 90).

Referenced by decaf::util::StlSet< Resource * >::equals().

6.516.3.6 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. 90).

6.516.3.7 template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable**< **E** > (p. 1207).

6.516.3.8 template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ( ) const [inline, virtual]

Implements **decaf::lang::Iterable**< **E** > (p. 1208).

6.516.3.9 template<typename E> virtual bool decaf::util::StlSet< E >::remove (const E & *value*) [inline, virtual]

Removes a single instance of the specified element from the collection.

More formally, removes an element e such that (value == NULL ? e == NULL : value == 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	The reference to the element to remove from this **Collection** (p. 660).

Returns

true if the collection was changed, false otherwise.

Exceptions

UnsupportedOperation-Exceptio	if this is an unmodifiable collection.
NullPointerException	if the **Collection** (p. 660) is a container of pointers and does not allow NULL values.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 92).

6.516.3.10 template<typename E> virtual int decaf::util::StlSet< E >::size ( ) const [inline, virtual]

Returns

The number of elements in this set.

Implements **decaf::util::Collection< E >** (p. 669).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlSet.h**

6.517 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**

6.517.1 Field Documentation

6.517.1.1 const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT [static]

6.517.1.2 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK [static]

6.517.1.3 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_AUTO [static]

6.517.1.4 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_CLIENT [static]

6.517.1.5 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_INDIVIDUAL [static]

6.517.1.6 const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN [static]

6.517.1.7 const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES [static]

6.517.1.8 const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT [static]

6.517.1.9 const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT [static]

6.517.1.10 const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED [static]

6.517.1.11 const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT [static]

6.517.1.12 const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD [static]

6.517.1.13 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK [static]

6.517.1.14 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID [static]

6.517.1.15 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY [static]

6.517.1.16 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONTENTLENGTH [static]

6.517.1.17 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CORRELATIONID [static]

6.517.1.18 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_DESTINATION [static]

6.517.1.19 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_DISPATCH_ASYNC [static]

6.517.1.20 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_EXCLUSIVE [static]

6.517.1.21 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_EXPIRES [static]

6.517.1.22 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ID [static]

6.517.1.23 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_JMSPRIORITY [static]

6.517.1.24 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_LOGIN [static]

6.517.1.25 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_MAXPENDINGMSGLIMIT [static]

6.517.1.26 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_MESSAGE [static]

6.517.1.27 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_MESSAGEID [static]

6.517.1.28 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_NOLOCAL [static]

6.517.1.29 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_OLDSUBSCRIPTIONNAME [static]

6.517.1.30 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_PASSWORD [static]

6.517.1.31 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_PERSISTENT** [static]

6.517.1.32 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_PREFETCHSIZE** [static]

6.517.1.33 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RECEIPT_REQUIRED** [static]

6.517.1.34 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RECEIPTID** [static]

6.517.1.35 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REDELIVERED** [static]

6.517.1.36 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REDELIVERYCOUNT** [static]

6.517.1.37 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REPLYTO** [static]

6.517.1.38 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REQUESTID** [static]

6.517.1.39 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RESPONSEID** [static]

6.517.1.40 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RETROACTIVE** [static]

6.517.1.41 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SELECTOR** [static]

6.517.1.42 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SESSIONID** [static]

6.517.1.43 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SUBSCRIPTION** [static]

6.517.1.44 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SUBSCRIPTIONNAME** [static]

6.517.1.45 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TIMESTAMP** [static]

6.517.1.46 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSACTIONID** [static]

6.517.1.47 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSFORMATION** [static]

6.517.1.48 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSFORMATION_ERROR** [static]

6.517.1.49 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TYPE** [static]

6.517.1.50 const std::string activemq::wireformat::stomp::StompCommandConstants::MESSAGE [static]

6.517.1.51 const std::string activemq::wireformat::stomp::StompCommandConstants::QUEUE_PREFIX [static]

6.517.1.52 const std::string activemq::wireformat::stomp::StompCommandConstants::RECEIPT [static]

6.517.1.53 const std::string activemq::wireformat::stomp::StompCommandConstants::SEND [static]

6.517.1.54 const std::string activemq::wireformat::stomp::StompCommandConstants::SUBSCRIBE [static]

6.517.1.55 const std::string activemq::wireformat::stomp::StompCommandConstants::TEMPQUEUE_PREFIX [static]

6.517.1.56 const std::string activemq::wireformat::stomp::StompCommandConstants::TEMPTOPIC_PREFIX [static]

6.517.1.57 const std::string activemq::wireformat::stomp::StompCommandConstants::TEXT [static]

6.517.1.58 const std::string activemq::wireformat::stomp::StompCommandConstants::TOPIC_PREFIX [static]

6.517.1.59 const std::string activemq::wireformat::stomp::StompCommandConstants::UNSUBSCRIBE [static]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompCommandConstants.h**

6.518 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**

  Clonse 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**

  Writes this Frame to an OuputStream in the Stomp Wire Format.
- void **fromStream** (**decaf::io::DataInputStream** *stream)

  Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

6.518.1 Detailed Description

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

6.518.2 Constructor & Destructor Documentation

6.518.2.1 activemq::wireformat::stomp::StompFrame::StompFrame ( )

Default constructor.

6.518.2.2 virtual activemq::wireformat::stomp::StompFrame::~StompFrame ( ) [virtual]

Destruction.

6.518.3 Member Function Documentation

6.518.3.1 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

6.518.3.2 void activemq::wireformat::stomp::StompFrame::copy (const StompFrame * *src*)

Copies the contents of the passed Frame to this one.

Parameters

src	- Frame to copy

6.518.3.3 void activemq::wireformat::stomp::StompFrame::fromStream (decaf::io::DataInputStream * *stream*)

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.

6.518.3.4 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

6.518.3.5 std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody ( ) [inline]

Non-const version of the body accessor.

6.518.3.6 std::size_t activemq::wireformat::stomp::StompFrame::getBodyLength ( ) const [inline]

Return the number of bytes contained in this frames body.

Returns

Body bytes length.

6.518.3.7 const std::string& activemq::wireformat::stomp::StompFrame::getCommand ( ) const [inline]

Accessor for this frame's command field.

6.518.3.8 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

6.518.3.9 const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ( ) const [inline]

6.518.3.10 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.

6.518.3.11 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.

6.518.3.12 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.

6.518.3.13 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.

6.518.3.14 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.

6.518.3.15 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.

6.518.3.16 void activemq::wireformat::stomp::StompFrame::toStream (decaf::io::DataOutputStream * *stream*) const

Writes this Frame to an OuputStream 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**

6.519 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from **StompFrame** (p. 2005)'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. 2005).*
- **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.

6.519.1 Detailed Description

Utility Methods used when marshaling to and from **StompFrame** (p. 2005)'s.

Since

3.0

6.519.2 Constructor & Destructor Documentation

6.519.2.1 activemq::wireformat::stomp::StompHelper::StompHelper ( )

6.519.2.2 virtual activemq::wireformat::stomp::StompHelper::~StompHelper ( ) [virtual]

6.519.3 Member Function Documentation

6.519.3.1 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.

6.519.3.2 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.

6.519.3.3 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.

6.519.3.4 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.

6.519.3.5 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.

6.519.3.6 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.

6.519.3.7 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.

6.519.3.8 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.

6.519.3.9 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.

6.519.3.10 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. 2005).

Parameters

message	- The message to move the Headers to.
frame	- The frame to extract headers from.

6.519.3.11 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.

6.519.3.12 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.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompHelper.h**

6.520 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)

  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)

  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. 2236) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer**
  < **transport::Transport** > **createNegotiator** (**const Pointer**< **transport::Transport** > &transport)

  If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

6.520.1 Constructor & Destructor Documentation

6.520.1.1 activemq::wireformat::stomp::StompWireFormat::StompWireFormat ( )

6.520.1.2 virtual activemq::wireformat::stomp::StompWireFormat::~StompWireFormat ( ) [virtual]

6.520.2 Member Function Documentation

6.520.2.1 virtual Pointer<transport::Transport> activemq::wireformat::stomp::StompWireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) [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. 2251).

Exceptions

UnsupportedOperation-Exception	if the **WireFormat** (p. 2236) doesn't have a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2237).

6.520.2.2 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. 2237).

6.520.2.3 virtual bool activemq::wireformat::stomp::StompWireFormat::hasNegotiator ( ) const [inline, virtual]

Returns true if this **WireFormat** (p. 2236) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 2236) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2237).

6.520.2.4 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. 2238).

6.520.2.5 virtual void activemq::wireformat::stomp::StompWireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport * *transport*, decaf::io::DataOutputStream * *out*) [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. 2238).

6.520.2.6 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. 2238).

6.520.2.7 virtual Pointer<commands::Command> activemq::wireformat::stomp::StompWireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) [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. 2238).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormat.h**

6.521 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)

  *Creates a new **WireFormat** (p. 2236) Object passing it a set of properties from which it can obtain any optional settings.*

6.521.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

6.521.2 Constructor & Destructor Documentation

6.521.2.1 activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory ( ) [inline]

6.521.2.2 virtual activemq::wireformat::stomp::StompWireFormatFactory::∼StompWireFormatFactory ( ) [inline, virtual]

6.521.3 Member Function Documentation

6.521.3.1 virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) [virtual]

Creates a new **WireFormat** (p. 2236) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties	- the Properties for this **WireFormat** (p. 2236)

Implements **activemq::wireformat::WireFormatFactory** (p. 2239).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormatFactory.h**

6.522 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

  Stops this service.

6.522.1 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

6.522.2 Constructor & Destructor Documentation

6.522.2.1 virtual cms::Stoppable::~Stoppable ( ) [virtual]

6.522.3 Member Function Documentation

6.522.3.1 virtual void cms::Stoppable::stop ( ) [pure virtual]

Stops this service.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs while stopping the Service.

Implemented in **activemq::core::ActiveMQConnection** (p. 164), **activemq::core::ActiveMQSession** (p. 274), **activemq::core::ActiveMQConsumer** (p. 189), **activemq::cmsutil::PooledSession** (p. 1632), and **activemq::cmsutil::CachedConsumer** (p. 572).

Referenced by activemq::cmsutil::PooledSession::stop().

The documentation for this class was generated from the following file:

- src/main/cms/**Stoppable.h**

6.523 decaf::util::logging::StreamHandler Class Reference

Stream based logging **Handler** (p. 1085).

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

Inheritance diagram for decaf::util::logging::StreamHandler:

Public Member Functions

- **StreamHandler** ()

  *Create a **StreamHandler** (p. 2017), with no current output stream.*
- **StreamHandler** (**decaf::io::OutputStream** ∗stream, **Formatter** ∗formatter)

  *Create a **StreamHandler** (p. 2017), with no current output stream.*
- virtual ∼**StreamHandler** ()
- virtual void **close** ()

  Close the current output stream.
- virtual void **flush** ()

  *Flush the **Handler** (p. 1085)'s output, clears any buffers.*
- virtual void **publish** (**const LogRecord** &record)

  *Publish the Log Record to this **Handler** (p. 1085).*
- virtual bool **isLoggable** (**const LogRecord** &record) **const**

  *Check if this **Handler** (p. 1085) would actually log a given **LogRecord** (p. 1333).*

Protected Member Functions

- virtual void **setOuputStream** (**decaf::io::OutputStream** ∗stream)

  Change the output stream.
- void **close** (bool closeStream)

  Closes this handler, but the underlying output stream is only closed if closeStream is true.

6.523.1 Detailed Description

Stream based logging **Handler** (p. 1085).

This is primarily intended as a base class or support class to be used in implementing other logging Handlers.

LogRecords are published to a given **`decaf::io::OutputStream`** (p. 1600).

Configuration: By default each **StreamHandler** (p. 2017) is initialized using the following **LogManager** (p. 1327) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

- decaf.util.logging.StreamHandler.level specifies the default level for the **Handler** (p. 1085) (defaults to **Level.-INFO** (p. 1256)).
  - decaf.util.logging.StreamHandler.filter specifies the name of a **Filter** (p. 1031) class to use (defaults to no **Filter** (p. 1031)).
  - decaf.util.logging.StreamHandler.formatter specifies the name of a **Formatter** (p. 1074) class to use (defaults to **decaf.util.logging.SimpleFormatter** (p. 1891)).

Since

1.0

6.523.2 Constructor & Destructor Documentation

6.523.2.1 decaf::util::logging::StreamHandler::StreamHandler ( )

Create a **StreamHandler** (p. 2017), with no current output stream.

6.523.2.2 decaf::util::logging::StreamHandler::StreamHandler (decaf::io::OutputStream * *stream,* Formatter * *formatter*)

Create a **StreamHandler** (p. 2017), with no current output stream.

6.523.2.3 virtual decaf::util::logging::StreamHandler::~StreamHandler ( ) [virtual]

6.523.3 Member Function Documentation

6.523.3.1 virtual void decaf::util::logging::StreamHandler::close ( ) [virtual]

Close the current output stream.

The close method will perform a flush and then close the **Handler** (p. 1085). After close has been called this **Handler** (p. 1085) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

Exceptions

IOException	if an I/O error occurs.

Implements **decaf::io::Closeable** (p. 633).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 769).

6.523.3.2 void decaf::util::logging::StreamHandler::close (bool *closeStream*) [protected]

Closes this handler, but the underlying output stream is only closed if closeStream is true.

Parameters

closeStream	whether to close the underlying output stream.

6.523.3.3 virtual void decaf::util::logging::StreamHandler::flush ( ) [virtual]

Flush the **Handler** (p. 1085)'s output, clears any buffers.

Implements **decaf::util::logging::Handler** (p. 1086).

6.523.3.4 virtual bool decaf::util::logging::StreamHandler::isLoggable (const LogRecord & *record*) const [virtual]

Check if this **Handler** (p. 1085) would actually log a given **LogRecord** (p. 1333).

Parameters

record	**LogRecord** (p. 1333) to check

Returns

true if the record can be logged with current settings.

Reimplemented from **decaf::util::logging::Handler** (p. 1087).

6.523.3.5 virtual void decaf::util::logging::StreamHandler::publish (const LogRecord & *record*) [virtual]

Publish the Log Record to this **Handler** (p. 1085).

Parameters

record	The **LogRecord** (p. 1333) to Publish

Implements **decaf::util::logging::Handler** (p. 1087).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 769).

6.523.3.6 virtual void decaf::util::logging::StreamHandler::setOuputStream (decaf::io::OutputStream * *stream*) [protected, virtual]

Change the output stream.

If there is a current output stream then the **Formatter** (p. 1074)'s tail string is written and the stream is flushed and closed. Then the output stream is replaced with the new output stream.

Parameters

stream	The new output stream. May not be NULL.

Exceptions

NullPointerException	if the passed stream is NULL.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

6.524 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 2020).

`#include <src/main/cms/StreamMessage.h>`

Inheritance diagram for cms::StreamMessage:

Public Member Functions

- virtual ~**StreamMessage** () throw ()
- virtual bool **readBoolean** () **const** =0

  Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value)=0

  Writes a boolean to the Stream message stream as a 1-byte value.
- virtual unsigned char **readByte** () **const** =0

Reads a Byte from the Stream message stream.

- virtual void **writeByte** (unsigned char value)=0

  Writes a byte to the Stream message stream as a 1-byte value.

- virtual int **readBytes** (std::vector< unsigned char > &value) **const** =0

  Reads a byte array from the Stream message stream.

- virtual void **writeBytes** (**const** std::vector< unsigned char > &value)=0

  Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

- virtual int **readBytes** (unsigned char *buffer, int length) **const** =0

  Reads a portion of the Stream message stream.

- virtual void **writeBytes** (**const** unsigned char *value, int offset, int length)=0

  Writes a portion of a byte array to the Stream message stream.

- virtual char **readChar** () **const** =0

  Reads a Char from the Stream message stream.

- virtual void **writeChar** (char value)=0

  Writes a char to the Stream message stream as a 1-byte value.

- virtual float **readFloat** () **const** =0

  Reads a 32 bit float from the Stream message stream.

- virtual void **writeFloat** (float value)=0

  Writes a float to the Stream message stream as a 4 byte value.

- virtual double **readDouble** () **const** =0

  Reads a 64 bit double from the Stream message stream.

- virtual void **writeDouble** (double value)=0

  Writes a double to the Stream message stream as a 8 byte value.

- virtual short **readShort** () **const** =0

  Reads a 16 bit signed short from the Stream message stream.

- virtual void **writeShort** (short value)=0

  Writes a signed short to the Stream message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () **const** =0

  Reads a 16 bit unsigned short from the Stream message stream.

- virtual void **writeUnsignedShort** (unsigned short value)=0

  Writes a unsigned short to the Stream message stream as a 2 byte value.

- virtual int **readInt** () **const** =0

  Reads a 32 bit signed integer from the Stream message stream.

- virtual void **writeInt** (int value)=0

  Writes a signed int to the Stream message stream as a 4 byte value.

- virtual long long **readLong** () **const** =0

  Reads a 64 bit long from the Stream message stream.

- virtual void **writeLong** (long long value)=0

  Writes a long long to the Stream message stream as a 8 byte value.

- virtual std::string **readString** () **const** =0

  Reads an ASCII String from the Stream message stream.

- virtual void **writeString** (**const** std::string &value)=0

  Writes an ASCII String to the Stream message stream.

6.524.1 Detailed Description

Interface for a **StreamMessage** (p. 2020).

The stream Messages provides a **Message** (p. 1426) 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. 2020) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 640). 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

6.524.2 Constructor & Destructor Documentation

6.524.2.1 virtual cms::StreamMessage::~StreamMessage ( ) throw () [virtual]

6.524.3 Member Function Documentation

6.524.3.1 virtual bool cms::StreamMessage::readBoolean ( ) const [pure virtual]

Reads a Boolean from the Stream message stream.

Returns

boolean value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 282).

6.524.3.2 virtual unsigned char cms::StreamMessage::readByte ( ) const [pure virtual]

Reads a Byte from the Stream message stream.

Returns

unsigned char value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 282).

6.524.3.3 virtual int cms::StreamMessage::readBytes (std::vector< unsigned char > & *value*) const [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. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 282).

6.524.3.4 virtual int cms::StreamMessage::readBytes (unsigned char * *buffer,* int *length*) const [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. 640) 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. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 283).

6.524.3.5 virtual char cms::StreamMessage::readChar ( ) const [pure virtual]

Reads a Char from the Stream message stream.

Returns

char value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 283).

6.524.3.6 virtual double cms::StreamMessage::readDouble ( ) const [pure virtual]

Reads a 64 bit double from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 284).

6.524.3.7 virtual float cms::StreamMessage::readFloat ( ) const [pure virtual]

Reads a 32 bit float from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 284).

6.524.3.8 virtual int cms::StreamMessage::readInt ( ) const [pure virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns

int value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.

MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 284).

6.524.3.9 virtual long long cms::StreamMessage::readLong ( ) const [pure virtual]

Reads a 64 bit long from the Stream message stream.

Returns

long long value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 285).

6.524.3.10 virtual short cms::StreamMessage::readShort ( ) const [pure virtual]

Reads a 16 bit signed short from the Stream message stream.

Returns

short value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 285).

6.524.3.11 virtual std::string cms::StreamMessage::readString ( ) const [pure virtual]

Reads an ASCII String from the Stream message stream.

Returns

String from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 285).

6.524.3.12 virtual unsigned short cms::StreamMessage::readUnsignedShort ( ) const [pure virtual]

Reads a 16 bit unsigned short from the Stream message stream.

Returns

unsigned short value from stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to read the message due to some internal error.
MessageEOFException *(p. 1477)*	- if unexpected end of message stream has been reached.
MessageFormatException *(p. 1478)*	- if this type conversion is invalid.
MessageNotReadable-Exception *(p. 1490)*	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 286).

6.524.3.13 virtual void cms::StreamMessage::writeBoolean (bool *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 286).

6.524.3.14 virtual void cms::StreamMessage::writeByte (unsigned char *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 287).

6.524.3.15 virtual void cms::StreamMessage::writeBytes (const std::vector< unsigned char > & *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 287).

6.524.3.16 virtual void cms::StreamMessage::writeBytes (const unsigned char * *value*, int *offset*, int *length*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 287).

6.524.3.17 virtual void cms::StreamMessage::writeChar (char *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 288).

6.524.3.18 virtual void cms::StreamMessage::writeDouble (double *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 288).

6.524.3.19 virtual void cms::StreamMessage::writeFloat (float *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 288).

6.524.3.20 virtual void cms::StreamMessage::writeInt (int *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 289).

6.524.3.21 virtual void cms::StreamMessage::writeLong (long long *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 289).

6.524.3.22 virtual void cms::StreamMessage::writeShort (short *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 289).

6.524.3.23 virtual void cms::StreamMessage::writeString (const std::string & *value*) [pure virtual]

Writes an ASCII String to the Stream message stream.

Parameters

value	String to write to the stream

Exceptions

CMSException *(p. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 289).

6.524.3.24 virtual void cms::StreamMessage::writeUnsignedShort (unsigned short *value*) [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. 640)*	- if the CMS provider fails to write the message due to some internal error.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 290).

The documentation for this class was generated from the following file:

- src/main/cms/**StreamMessage.h**

6.525 decaf::lang::String Class Reference

The **String** (p. 2031) class represents an immutable sequence of chars.

`#include <src/main/decaf/lang/String.h>`

Inheritance diagram for decaf::lang::String:

Public Member Functions

- **String** ()

  *Creates a new empty **String** (p. 2031) object.*
- **String** (**const String** &source)

  *Create a new **String** (p. 2031) object that represents the given STL string.*
- **String** (**const** std::string &source)

  *Create a new **String** (p. 2031) object that represents the given STL string.*
- **String** (**const** char *array, int size)

  *Create a new **String** (p. 2031) object that represents the given array of characters.*
- **String** (**const** char *array, int size, int offset, int **length**)

  *Create a new **String** (p. 2031) object that represents the given array of characters.*
- virtual ~**String** ()
- **String** & **operator=** (**const String** &)
- **String** & **operator=** (**const** std::string &)

- bool **isEmpty** () **const**
- virtual int **length** () **const**

  Returns

  the length of the underlying character sequence.
- virtual char **charAt** (int index) **const**

  Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

  Parameters

index	*The position to return the char at.*

  Returns

  the char at the given position.

  Exceptions

IndexOutOfBoundsException	*if index is > than* ***length()*** *(p. 624) or negative*

- virtual **CharSequence** * **subSequence** (int start, int end) **const**

  Returns a new ***CharSequence*** *(p. 623) 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. 623)*

  Exceptions

IndexOutOfBoundsException	*if start or end >* ***length()*** *(p. 624) or start or end are negative.*

- virtual std::string **toString** () **const**

  Returns

  the string representation of this ***CharSequence*** *(p. 623)*

Static Public Member Functions

- static **String valueOf** (bool value)

  Returns a ***String*** *(p. 2031) that represents the value of the given boolean value.*
- static **String valueOf** (char value)

  Returns a ***String*** *(p. 2031) that represents the value of the given char value.*
- static **String valueOf** (float value)

  Returns a ***String*** *(p. 2031) that represents the value of the given float value.*
- static **String valueOf** (double value)

  Returns a ***String*** *(p. 2031) that represents the value of the given double value.*
- static **String valueOf** (short value)

  Returns a ***String*** *(p. 2031) that represents the value of the given short value.*
- static **String valueOf** (int value)

  Returns a ***String*** *(p. 2031) that represents the value of the given integer value.*
- static **String valueOf** (long long value)

  Returns a ***String*** *(p. 2031) that represents the value of the given 64bit long value.*

6.525.1 Detailed Description

The **String** (p. 2031) class represents an immutable sequence of chars.

Since

1.0

6.525.2 Constructor & Destructor Documentation

6.525.2.1 decaf::lang::String::String ( )

Creates a new empty **String** (p. 2031) object.

6.525.2.2 decaf::lang::String::String (const String & *source*)

Create a new **String** (p. 2031) object that represents the given STL string.

Parameters

source	The string to copy into this new **String** (p. 2031) object.

6.525.2.3 decaf::lang::String::String (const std::string & *source*)

Create a new **String** (p. 2031) object that represents the given STL string.

Parameters

source	The string to copy into this new **String** (p. 2031) object.

6.525.2.4 decaf::lang::String::String (const char * *array,* int *size*)

Create a new **String** (p. 2031) object that represents the given array of characters.

The method takes the size of the array as a parameter to allow for strings that are not NULL terminated, the caller can pass strlen(array) in the case where the array is properly NULL terminated.

Parameters

array	The character buffer to copy into this new **String** (p. 2031) object.
size	The size of the string buffer given, in case the string is not NULL terminated.

Exceptions

NullPointerException	if the character array parameter is NULL.
IndexOutOfBounds-Exception	if the size parameter is negative.

6.525.2.5 decaf::lang::String::String (const char * *array,* int *size,* int *offset,* int *length*)

Create a new **String** (p. 2031) object that represents the given array of characters.

The method takes the size of the array as a parameter to allow for strings that are not NULL terminated, the caller can pass strlen(array) in the case where the array is properly NULL terminated.

Parameters

array	The character buffer to copy into this new **String** (p. 2031) object.
size	The size of the string buffer given, in case the string is not NULL terminated.

offset	The position to start copying from in the given buffer.
length	The number of bytes to copy from the given buffer starting from the offset.

Exceptions

NullPointerException	if the character array parameter is NULL.
IndexOutOfBounds-Exception	if the size, offset or length parameter is negative or if the length to copy is greater than the span of size - offset.

6.525.2.6 virtual decaf::lang::String::~String ( ) [virtual]

6.525.3 Member Function Documentation

6.525.3.1 virtual char decaf::lang::String::charAt (int *index*) const [virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

index	The position to return the char at.

Returns

the char at the given position.

Exceptions

IndexOutOfBounds-Exception	if index is $>$ than **length()** (p. 624) or negative

Implements **decaf::lang::CharSequence** (p. 623).

6.525.3.2 bool decaf::lang::String::isEmpty ( ) const

Returns

true if the length of this **String** (p. 2031) is zero.

6.525.3.3 virtual int decaf::lang::String::length ( ) const [virtual]

Returns

the length of the underlying character sequence.

Implements **decaf::lang::CharSequence** (p. 624).

6.525.3.4 String& decaf::lang::String::operator= (const String &)

6.525.3.5 String& decaf::lang::String::operator= (const std::string &)

6.525.3.6 virtual CharSequence* decaf::lang::String::subSequence (int *start,* int *end*) const [virtual]

Returns a new **CharSequence** (p. 623) 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. 623)

Exceptions

IndexOutOfBounds-Exception	if start or end $>$ **length()** (p. 624) or start or end are negative.

Implements **decaf::lang::CharSequence** (p. 624).

6.525.3.7 virtual std::string decaf::lang::String::toString ( ) const [virtual]

Returns

the string representation of this **CharSequence** (p. 623)

Implements **decaf::lang::CharSequence** (p. 624).

6.525.3.8 static String decaf::lang::String::valueOf (bool *value*) [static]

Returns a **String** (p. 2031) that represents the value of the given boolean value.

Parameters

value	The value whose string representation is to be returned.

Returns

"true" if the boolean is true, "false" otherwise.

6.525.3.9 static String decaf::lang::String::valueOf (char *value*) [static]

Returns a **String** (p. 2031) that represents the value of the given char value.

Parameters

value	The value whose string representation is to be returned.

Returns

a **String** (p. 2031) that contains the single character value given.

6.525.3.10 static String decaf::lang::String::valueOf (float *value*) [static]

Returns a **String** (p. 2031) that represents the value of the given float value.

Parameters

value	The value whose string representation is to be returned.

Returns

a **String** (p. 2031) that contains the string representation of the float value given.

6.525.3.11 static String decaf::lang::String::valueOf (double *value*) [static]

Returns a **String** (p. 2031) that represents the value of the given double value.

Parameters

value	The value whose string representation is to be returned.

Returns

a **String** (p. 2031) that contains the string representation of the double value given.

6.525.3.12 static String decaf::lang::String::valueOf (short *value*) [static]

Returns a **String** (p. 2031) that represents the value of the given short value.

Parameters

value	The value whose string representation is to be returned.

Returns

a **String** (p. 2031) that contains the string representation of the short value given.

6.525.3.13 static String decaf::lang::String::valueOf (int *value*) [static]

Returns a **String** (p. 2031) that represents the value of the given integer value.

Parameters

value	The value whose string representation is to be returned.

Returns

a **String** (p. 2031) that contains the string representation of the integer value given.

6.525.3.14 static String decaf::lang::String::valueOf (long long *value*) [static]

Returns a **String** (p. 2031) that represents the value of the given 64bit long value.

Parameters

value	The value whose string representation is to be returned.

Returns

a **String** (p. 2031) that contains the string representation of the 64 bit long value given.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**String.h**

6.526 decaf::util::StringTokenizer Class Reference

Class that allows for parsing of string based on Tokens.

`#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** ()

  Returns the next token from this string tokenizer.
- virtual std::string **nextToken** (**const** std::string &delim)

  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.

6.526.1 Detailed Description

Class that allows for parsing of string based on Tokens.

Since

1.0

6.526.2 Constructor & Destructor Documentation

6.526.2.1 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. 2036) 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

6.526.2.2 virtual decaf::util::StringTokenizer::~StringTokenizer ( ) [virtual]

6.526.3 Member Function Documentation

6.526.3.1 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

6.526.3.2 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

6.526.3.3 virtual std::string decaf::util::StringTokenizer::nextToken ( ) [virtual]

Returns the next token from this string tokenizer.

Returns

string value of next token

Exceptions

NoSuchElementException *(p. 1537)*	if there are no more tokens in this string.

6.526.3.4 virtual std::string decaf::util::StringTokenizer::nextToken (const std::string & *delim*) [virtual]

Returns the next token in this string tokenizer's string.

First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 2036) 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	The string containing the new set of delimiters.

Returns

next string in the token list

Exceptions

NoSuchElementException *(p. 1537)*	if there are no more tokens in this string.

6.526.3.5 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 tokenizer 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

6.526.3.6 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**

6.527 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **SubscriptionInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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 & **getSubcriptionName** () **const**
- virtual std::string & **getSubcriptionName** ()
- virtual void **setSubcriptionName** (**const** std::string &**subcriptionName**)
- virtual **const Pointer**
  < **ActiveMQDestination** > & **getSubscribedDestination** () **const**
- virtual **Pointer**
  < **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (**const Pointer**< **ActiveMQDestination** > &**subscribed-Destination**)

Static Public Attributes

- static **const** unsigned char **ID_SUBSCRIPTIONINFO** = 55

Protected Attributes

- std::string **clientId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::string **selector**
- std::string **subcriptionName**
- **Pointer**< **ActiveMQDestination** > **subscribedDestination**

6.527.1 Constructor & Destructor Documentation

6.527.1.1 activemq::commands::SubscriptionInfo::SubscriptionInfo ( )

6.527.1.2 virtual activemq::commands::SubscriptionInfo::~SubscriptionInfo ( ) [virtual]

6.527.2 Member Function Documentation

6.527.2.1 virtual SubscriptionInfo* activemq::commands::SubscriptionInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.527.2.2 virtual void activemq::commands::SubscriptionInfo::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. 879).

6.527.2.3 virtual bool activemq::commands::SubscriptionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

6.527.2.4 virtual const std::string& activemq::commands::SubscriptionInfo::getClientId ( ) const [virtual]

6.527.2.5 virtual std::string& activemq::commands::SubscriptionInfo::getClientId ( ) [virtual]

6.527.2.6 virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.527.2.7 virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ( ) const [virtual]

6.527.2.8 virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ( ) [virtual]

6.527.2.9 virtual const std::string& activemq::commands::SubscriptionInfo::getSelector ( ) const [virtual]

6.527.2.10 virtual std::string& activemq::commands::SubscriptionInfo::getSelector ( ) [virtual]

6.527.2.11 virtual const std::string& activemq::commands::SubscriptionInfo::getSubcriptionName ( ) const [virtual]

6.527.2.12 virtual std::string& activemq::commands::SubscriptionInfo::getSubcriptionName ( ) [virtual]

6.527.2.13 virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ( ) const [virtual]

6.527.2.14 virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ( ) [virtual]

6.527.2.15 virtual void activemq::commands::SubscriptionInfo::setClientId (const std::string & *clientId*) [virtual]

6.527.2.16 virtual void activemq::commands::SubscriptionInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.527.2.17 virtual void activemq::commands::SubscriptionInfo::setSelector (const std::string & *selector*) [virtual]

6.527.2.18 virtual void activemq::commands::SubscriptionInfo::setSubcriptionName (const std::string & *subcriptionName*) [virtual]

6.527.2.19 virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination (const Pointer< ActiveMQDestination > & *subscribedDestination*) [virtual]

6.527.2.20 virtual std::string activemq::commands::SubscriptionInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

6.527.3 Field Documentation

6.527.3.1 std::string activemq::commands::SubscriptionInfo::clientId [protected]

6.527.3.2 Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination [protected]

6.527.3.3 const unsigned char activemq::commands::SubscriptionInfo::ID_SUBSCRIPTIONINFO = 55 [static]

6.527.3.4 std::string activemq::commands::SubscriptionInfo::selector [protected]

6.527.3.5 std::string activemq::commands::SubscriptionInfo::subcriptionName [protected]

6.527.3.6 Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**SubscriptionInfo.h**

6.528 activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2042).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/SubscriptionInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual ~**SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.528.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2042).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.528.2 Constructor & Destructor Documentation

6.528.2.1 activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller ( ) [inline]

6.528.2.2 virtual activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::~SubscriptionInfoMarshaller ( ) [inline, virtual]

6.528.3 Member Function Documentation

6.528.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.528.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.528.3.3 virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.528.3.4 virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *format,* commands::DataStructure * *command,* decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.528.3.5 virtual int activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.528.3.6 virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.528.3.7 virtual void activemq::wireformat::openwire::marshal::generated::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**SubscriptionInfoMarshaller.h**

6.529 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 ~**Synchronizable** ()
- virtual void **lock** ()=0

  Locks the object.
- virtual bool **tryLock** ()=0

  *Attempts to **Lock** (p. 1304) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0

  Unlocks the object.
- virtual void **wait** ()=0

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)=0

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)=0

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()=0

  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()=0

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

6.529.1 Detailed Description

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

Since

1.0

6.529.2 Constructor & Destructor Documentation

6.529.2.1 virtual decaf::util::concurrent::Synchronizable::~Synchronizable ( ) [inline, virtual]

6.529.3 Member Function Documentation

6.529.3.1 virtual void decaf::util::concurrent::Synchronizable::lock ( ) [pure virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 808), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 808), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 708), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 708), **decaf::util::AbstractCollection< E >** (p. 91), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 91), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 91), **decaf::util::AbstractCollection< Resource * >** (p. 91), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 91), **decaf::util::AbstractCollection< CompositeTask * >** (p. 91), **decaf::util::AbstractCollection< URI >** (p. 91), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 91), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 91), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 91), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 91), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 91), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 91), **decaf::util::AbstractCollection< cms::Destination * >** (p. 91), **decaf::util::AbstractCollection< cms::Session * >** (p. 91), **decaf::util::AbstractCollection< cms::Connection * >** (p. 91), **decaf::io::InputStream** (p. 1137), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1984), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1984), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1984), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1984), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1984), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1984), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1984), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1984), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1984), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1984), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1984), **decaf::util::StlQueue< T >** (p. 1991), **decaf::io::OutputStream** (p. 1603), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1896), **activemq::core::FifoMessageDispatchChannel** (p. 1026), **decaf::util::concurrent::Mutex** (p. 1520), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2054).

6.529.3.2 virtual void decaf::util::concurrent::Synchronizable::notify ( ) [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. 2045) Object.
RuntimeException	if an error occurs while notifying one of the waiting threads.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 808), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 808), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 709), **decaf::util::AbstractCollection< E >** (p. 91), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 91), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 91), **decaf::util::AbstractCollection< Resource * >** (p. 91), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 91), **decaf::util::AbstractCollection< CompositeTask * >** (p. 91), **decaf::util::AbstractCollection< URI >** (p. 91), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 91), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 91), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 91), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 91), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 91), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 91), **decaf::util::AbstractCollection< cms::Destination * >** (p. 91), **decaf::util::AbstractCollection< cms::Session * >** (p. 91), **decaf::util::AbstractCollection< cms::Connection * >** (p. 91), **decaf::io::InputStream** (p. 1137), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1984), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1984), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1984), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1984), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1984), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1984), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1984), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1984), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1984), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1984), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1984), **decaf::util::StlQueue< T >** (p. 1991), **decaf::io::OutputStream** (p. 1603), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1897), **activemq::core::FifoMessageDispatchChannel** (p. 1026), **decaf::util::concurrent::Mutex** (p. 1520), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2054).

6.529.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll ( ) [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. 2045) Object.
RuntimeException	if an error occurs while notifying the waiting threads.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 808), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 808), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 709), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 709), **decaf::util::AbstractCollection< E >** (p. 91), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 91), **decaf::util::AbstractCollection< Pointer< Synchronization**

> > (p. 91), **decaf::util::AbstractCollection< Resource * >** (p. 91), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 91), **decaf::util::AbstractCollection< CompositeTask * >** (p. 91), **decaf::util::AbstractCollection< URI >** (p. 91), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 91), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 91), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 91), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 91), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 91), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 91), **decaf::util::AbstractCollection< cms::Destination * >** (p. 91), **decaf::util::AbstractCollection< cms::Session * >** (p. 91), **decaf::util::AbstractCollection< cms::Connection * >** (p. 91), **decaf::io::InputStream** (p. 1138), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1984), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1984), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1984), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1984), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1984), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1984), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1984), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1984), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1984), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1984), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1984), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1984), **decaf::util::StlQueue< T >** (p. 1992), **decaf::io::OutputStream** (p. 1603), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1897), **activemq::core::FifoMessageDispatchChannel** (p. 1026), **decaf::util::concurrent::Mutex** (p. 1520), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2054).

6.529.3.4 virtual bool decaf::util::concurrent::Synchronizable::tryLock ( ) [pure virtual]

Attempts to **Lock** (p. 1304) 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 **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 811), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 811), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 713), **decaf::util::AbstractCollection< E >** (p. 94), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 94), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 94), **decaf::util::AbstractCollection< Resource * >** (p. 94), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 94), **decaf::util::AbstractCollection< CompositeTask * >** (p. 94), **decaf::util::AbstractCollection< URI >** (p. 94), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 94), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 94), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 94), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 94), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 94), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 94), **decaf::util::AbstractCollection< cms::Destination * >** (p. 94), **decaf::util::AbstractCollection< cms::Session * >** (p. 94), **decaf::util::AbstractCollection< cms::Connection * >** (p. 94), **decaf::io::InputStream** (p. 1141), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1986), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1986), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1986), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask ***

> (p. 1986), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1986), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1986), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1986), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1986), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1986), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1986), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1986), **decaf::util::StlQueue< T >** (p. 1993), **decaf::io::OutputStream** (p. 1604), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1898), **activemq::core::FifoMessageDispatchChannel** (p. 1027), **decaf::util::concurrent::Mutex** (p. 1521), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2054).

6.529.3.5 virtual void decaf::util::concurrent::Synchronizable::unlock ( ) [pure virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 812), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 812), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 713), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 713), **decaf::util::AbstractCollection< E >** (p. 94), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 94), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 94), **decaf::util::AbstractCollection< Resource * >** (p. 94), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 94), **decaf::util::AbstractCollection< CompositeTask * >** (p. 94), **decaf::util::AbstractCollection< URI >** (p. 94), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 94), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 94), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 94), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 94), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 94), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 94), **decaf::util::AbstractCollection< cms::Destination * >** (p. 94), **decaf::util::AbstractCollection< cms::Session * >** (p. 94), **decaf::util::AbstractCollection< cms::Connection * >** (p. 94), **decaf::io::InputStream** (p. 1141), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1986), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1986), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1986), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1986), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1986), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1986), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1986), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1986), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1986), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1986), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1986), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1986), **decaf::util::StlQueue< T >** (p. 1993), **decaf::io::OutputStream** (p. 1604), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1898), **activemq::core::FifoMessageDispatchChannel** (p. 1028), **decaf::util::concurrent::Mutex** (p. 1521), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2055).

6.529.3.6 virtual void decaf::util::concurrent::Synchronizable::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 812), **decaf::util::concurrent::-CopyOnWriteArrayList< ServiceListener * >** (p. 812), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 714), **decaf::util-::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::-COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 714), **decaf::util::AbstractCollection< E >** (p. 95), **decaf::util::Abstract-Collection< Pointer< Transport > >** (p. 95), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 95), **decaf::util::AbstractCollection< Resource * >** (p. 95), **decaf::util::AbstractCollection< cms-::MessageConsumer * >** (p. 95), **decaf::util::AbstractCollection< CompositeTask * >** (p. 95), **decaf-::util::AbstractCollection< URI >** (p. 95), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 95), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 95), **decaf::util::Abstract-Collection< PrimitiveValueNode >** (p. 95), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 95), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 95), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 95), **decaf::util::AbstractCollection< cms::Destination * >** (p. 95), **decaf-::util::AbstractCollection< cms::Session * >** (p. 95), **decaf::util::AbstractCollection< cms::Connection * >** (p. 95), **decaf::io::InputStream** (p. 1141), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1987), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1987), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1987), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1987), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1987), **decaf::util::StlMap< std-::string, cms::Queue * >** (p. 1987), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1987), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::Consumer-Id::COMPARATOR >** (p. 1987), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1987), **decaf::util-::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1987), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1987), **decaf::util::StlMap< std::string, Cached-Producer * >** (p. 1987), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1987), **decaf::util::StlQueue< T >** (p. 1993), **decaf::io::OutputStream** (p. 1604), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1898), **activemq::core::FifoMessageDispatchChannel** (p. 1028), **decaf::util::concurrent::Mutex** (p. 1521), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2055).

6.529.3.7 virtual void decaf::util::concurrent::Synchronizable::wait (long long *millisecs*) [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

millisecs	the time in milliseconds to wait, or WAIT_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the **Synchronizable** (p. 2045) Object.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 812), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 812), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 714), **decaf::util::AbstractCollection< E >** (p. 95), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 95), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 95), **decaf::util::AbstractCollection< Resource * >** (p. 95), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 95), **decaf::util::AbstractCollection< CompositeTask * >** (p. 95), **decaf::util::AbstractCollection< URI >** (p. 95), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 95), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 95), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 95), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 95), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 95), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 95), **decaf::util::AbstractCollection< cms::Destination * >** (p. 95), **decaf::util::AbstractCollection< cms::Session * >** (p. 95), **decaf::util::AbstractCollection< cms::Connection * >** (p. 95), **decaf::io::InputStream** (p. 1142), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1987), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1987), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1987), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1987), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1987), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1987), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1987), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1987), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1987), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1987), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1987), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1987), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1987), **decaf::util::StlQueue< T >** (p. 1994), **decaf::io::OutputStream** (p. 1604), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1899), **activemq::core::FifoMessageDispatchChannel** (p. 1028), **decaf::util::concurrent::Mutex** (p. 1522), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2055).

6.529.3.8 virtual void decaf::util::concurrent::Synchronizable::wait (long long *millisecs*, int *nanos*) [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_INIFINITE
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. 2045) Object.

Implemented in **decaf::util::concurrent::CopyOnWriteArrayList< E >** (p. 813), **decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >** (p. 813), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 714), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 714), **decaf::util::AbstractCollection< E >** (p. 95), **decaf::util::AbstractCollection< Pointer< Transport > >** (p. 95), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 95), **decaf::util::AbstractCollection< Resource * >** (p. 95), **decaf::util::AbstractCollection< cms::MessageConsumer * >** (p. 95), **decaf::util::AbstractCollection< CompositeTask * >** (p. 95), **decaf::util::AbstractCollection< URI >** (p. 95), **decaf::util::AbstractCollection< Pointer< MessageDispatch > >** (p. 95), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 95), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 95), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 95), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 95), **decaf::util::AbstractCollection< cms::MessageProducer * >** (p. 95), **decaf::util::AbstractCollection< cms::Destination * >** (p. 95), **decaf::util::AbstractCollection< cms::Session * >** (p. 95), **decaf::util::AbstractCollection< cms::Connection * >** (p. 95), **decaf::io::InputStream** (p. 1142), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 1987), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 1987), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 1987), **decaf::util::StlMap< decaf::lang::Runnable *, decaf::util::TimerTask * >** (p. 1987), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 1987), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 1987), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 1987), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 1987), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 1987), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 1987), **decaf::util::StlMap< int, Pointer< Command > >** (p. 1987), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 1987), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 1987), **decaf::util::StlQueue< T >** (p. 1994), **decaf::io::OutputStream** (p. 1605), **activemq::core::SimplePriorityMessageDispatchChannel** (p. 1899), **activemq::core::FifoMessageDispatchChannel** (p. 1028), **decaf::util::concurrent::Mutex** (p. 1522), and **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2055).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Synchronizable.h**

6.530 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 ~**SynchronizableImpl** ()
- virtual void **lock** ()

  Locks the object.
- virtual bool **tryLock** ()

  Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** ()

  Unlocks the object.
- virtual void **wait** ()

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)

  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()

  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()

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

6.530.1 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

6.530.2 Constructor & Destructor Documentation

6.530.2.1 decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl ( )

6.530.2.2 virtual decaf::internal::util::concurrent::SynchronizableImpl::~SynchronizableImpl ( ) [virtual]

6.530.3 Member Function Documentation

6.530.3.1 virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock ( ) [virtual]

Locks the object.

Exceptions

RuntimeException	if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2046).

6.530.3.2 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify ( ) [virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2047).

6.530.3.3 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notifyAll ( ) [virtual]

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

Must have this object locked before calling.

Exceptions

IllegalMonitorState-Exception	- 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. 2048).

6.530.3.4 virtual bool decaf::internal::util::concurrent::SynchronizableImpl::tryLock ( ) [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. 2048).

6.530.3.5 virtual void decaf::internal::util::concurrent::SynchronizableImpl::unlock ( ) [virtual]

Unlocks the object.

Exceptions

RuntimeException	if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2049).

6.530.3.6 virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait ( ) [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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2050).

6.530.3.7 virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long *millisecs*) [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_INIFINITE

Exceptions

RuntimeException	if an error occurs while waiting on the object.
InterruptedException	if the wait is interrupted before it completes.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2051).

6.530.3.8 virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long *millisecs,* int *nanos*) [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_INIFINITE
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.
IllegalMonitorState-Exception	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2052).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**SynchronizableImpl.h**

6.531 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 2056), 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** ()=0
- virtual void **afterCommit** ()=0
- virtual void **afterRollback** ()=0

6.531.1 Detailed Description

Transacted Object **Synchronization** (p. 2056), used to sync the events of a Transaction with the items in the Transaction.

6.531.2 Constructor & Destructor Documentation

6.531.2.1 virtual activemq::core::Synchronization::~Synchronization ( ) [inline, virtual]

6.531.3 Member Function Documentation

6.531.3.1 virtual void activemq::core::Synchronization::afterCommit ( ) [pure virtual]

6.531.3.2 virtual void activemq::core::Synchronization::afterRollback ( ) [pure virtual]

6.531.3.3 virtual void activemq::core::Synchronization::beforeEnd ( ) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Synchronization.h**

6.532 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

A **blocking queue** (p. 417) 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>`

Inheritance diagram for decaf::util::concurrent::SynchronousQueue< E >:

Data Structures

- class **EmptyIterator**

Public Member Functions

- **SynchronousQueue** ()
- virtual ~**SynchronousQueue** ()
- virtual void **put** (**const** E &value)

  Adds the specified element to this queue, waiting if necessary for another thread to receive it.
- virtual bool **offer** (**const** E &e, long long timeout, **const TimeUnit** &unit)

  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)

Inserts the specified element into this queue, if another thread is waiting to receive it.

- virtual E **take** ()

  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)

  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**

  *Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.*

- virtual **decaf::util::Iterator** < E > * **iterator** ()
- virtual **decaf::util::Iterator** < E > * **iterator** () **const**
- virtual bool **isEmpty** () **const**

  Returns true if this collection contains no elements.

- virtual int **size** () **const**

  Returns the number of elements in this collection.

- virtual int **remainingCapacity** () **const**

  Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

- virtual void **clear** ()

  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. 1210) 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

UnsupportedOperation-Exception	*if the clear operation is not supported by this collection*

This implementation repeatedly invokes poll until it returns false.

- virtual bool **contains** (**const** E &value **DECAF_UNUSED**) **const**
- virtual bool **containsAll** (**const Collection**< E > &collection) **const**

  Returns true if this collection contains all of the elements in the specified collection.

  Parameters

collection	*The **Collection** (p. 660) to compare to this one.*

  Exceptions

NullPointerException	*if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).*

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.

- virtual bool **remove** (**const** E &value **DECAF_UNUSED**)
- virtual bool **removeAll** (**const Collection**< E > &collection **DECAF_UNUSED**)
- virtual bool **retainAll** (**const Collection**< E > &collection **DECAF_UNUSED**)
- virtual bool **peek** (E &result **DECAF_UNUSED**) **const**
- virtual std::vector< E > **toArray** () **const**

  *Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).*

- virtual int **drainTo** (**Collection**< E > &c)

  Removes all available elements from this queue and adds them to the given collection.

- virtual int **drainTo** (**Collection**< E > &c, int maxElements)

Removes at most the given number of available elements from this queue and adds them to the given collection.

6.532.1 Detailed Description

template<typename E>class decaf::util::concurrent::SynchronousQueue< E >

A **blocking queue** (p. 417) 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. 2062) will return `null`. For purposes of other **`Collection`** (p. 660) methods (for example `contains`), a **`SynchronousQueue`** (p. 2057) 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. 660) and **Iterator** (p. 1209) interfaces.

Since

1.0

6.532.2 Constructor & Destructor Documentation

6.532.2.1 template<typename E > decaf::util::concurrent::SynchronousQueue< E >::SynchronousQueue ( ) [inline]

6.532.2.2 template<typename E > virtual decaf::util::concurrent::SynchronousQueue< E >::~SynchronousQueue ( ) [inline, virtual]

6.532.3 Member Function Documentation

6.532.3.1 template<typename E > virtual void decaf::util::concurrent::SynchronousQueue< E >::clear ( ) [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. 1210) 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

UnsupportedOperation-Exception	if the clear operation is not supported by this collection

This implementation repeatedly invokes poll until it returns false.

This implementation repeatedly invokes poll until it returns false.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 110).

6.532.3.2 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::contains (const E &value *DECAF_UNUSED*) const [inline, virtual]

6.532.3.3 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::containsAll (const Collection< E > & *collection*) const [inline, virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection	The **Collection** (p. 660) to compare to this one.

Exceptions

NullPointerException	if the **Collection** (p. 660) contains pointers and the **Collection** (p. 660) does not allow for NULL elements (optional check).

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.

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.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 89).

References decaf::util::Collection< E >::isEmpty().

6.532.3.4 template<typename E > virtual int decaf::util::concurrent::SynchronousQueue< E >::drainTo (Collection< E > & *c*) [inline, virtual]

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c	the collection to transfer elements into

Returns

the number of elements transferred

Exceptions

UnsupportedOperation-Exception	if addition of elements is not supported by the specified collection
IllegalArgumentException	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 419).

References decaf::util::Collection< E >::add(), decaf::util::AbstractQueue< E >::element(), and decaf::util::concurrent::SynchronousQueue< E >::poll().

6.532.3.5 template<typename E > virtual int decaf::util::concurrent::SynchronousQueue< E >::drainTo (Collection< E > & *c,* int *maxElements*) [inline, virtual]

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

Parameters

c	the collection to transfer elements into
maxElements	the maximum number of elements to transfer

Returns

the number of elements transferred

Exceptions

UnsupportedOperationException	if addition of elements is not supported by the specified collection
IllegalArgumentException	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 420).

References decaf::util::Collection< E >::add(), decaf::util::AbstractQueue< E >::element(), and decaf::util::concurrent::SynchronousQueue< E >::poll().

6.532.3.6 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::equals (const Collection< E > & *collection*) const [inline, virtual]

Answers true if this **Collection** (p. 660) and the one given are the same size and if each element contained in the **Collection** (p. 660) given is equal to an element contained in this collection.

Parameters

collection	- The **Collection** (p. 660) to be compared to this one.

Returns

true if this **Collection** (p. 660) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

6.532.3.7 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::isEmpty ( ) const [inline, virtual]

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 2064) == 0.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 90).

6.532.3.8 template<typename E > virtual decaf::util::Iterator<E>* decaf::util::concurrent::SynchronousQueue< E >::iterator ( ) [inline, virtual]

Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1207).

6.532.3.9 template<typename E > virtual decaf::util::Iterator<E>* decaf::util::concurrent::SynchronousQueue< E >::iterator ( ) const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1208).

6.532.3.10 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::offer (const E & *e,* long long *timeout,* const TimeUnit & *unit*) [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	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
NullPointerException	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
IllegalArgumentException	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 420).

6.532.3.11 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::offer (const E & *value*) [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. 1723)

Returns

`true` if the element was added to this queue, else `false`

Exceptions

NullPointerException	if the **Queue** (p. 1723) implementation does not allow Null values to be inserted into the **Queue** (p. 1723).
IllegalArgumentException	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 1724).

6.532.3.12 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::peek (E &result *DECAF_UNUSED*) const [inline, virtual]

6.532.3.13 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::poll (E & *result*, long long *timeout*, const TimeUnit & *unit*) [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. 1723) 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. 1723) was copied to the result param or false if no value could be returned.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 421).

Referenced by decaf::util::concurrent::SynchronousQueue< E >::drainTo().

6.532.3.14 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. 1723) should be copied to.

Returns

true if the head of the **Queue** (p. 1723) was copied to the result param or false if no value could be returned.

Implements **decaf::util::Queue< E >** (p. 1725).

6.532.3.15 template<typename E > virtual void decaf::util::concurrent::SynchronousQueue< E >::put (const E & *value*) [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. 1723).

Exceptions

InterruptedException	Inserts the specified element into this queue, waiting if necessary for space to become available.
NullPointerException	Inserts the specified element into this queue, waiting if necessary for space to become available.
IllegalArgumentException	Inserts the specified element into this queue, waiting if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 421).

6.532.3.16 template<typename E > virtual int decaf::util::concurrent::SynchronousQueue< E >::remainingCapacity ( ) const [inline, virtual]

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

Returns

the remaining capacity

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 421).

6.532.3.17 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::remove (const E &value *DECAF_UNUSED*) [inline, virtual]

6.532.3.18 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::removeAll (const Collection< E > &collection *DECAF_UNUSED*) [inline, virtual]

6.532.3.19 template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::retainAll (const Collection< E > &collection *DECAF_UNUSED*) [inline, virtual]

6.532.3.20 template<typename E > virtual int decaf::util::concurrent::SynchronousQueue< 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. 669).

6.532.3.21 template<typename E > virtual E decaf::util::concurrent::SynchronousQueue< E >::take ( ) [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	Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 422).

6.532.3.22 template<typename E > virtual std::vector<E> decaf::util::concurrent::SynchronousQueue< E >::toArray ( ) const [inline, virtual]

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 660).

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. 660)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 94).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**SynchronousQueue.h**

6.533 decaf::lang::System Class Reference

The **System** (p. 2065) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

`#include <src/main/decaf/lang/System.h>`

Public Member Functions

- virtual ~**System** ()

Static Public Member Functions

- static void **arraycopy** (**const** char *src, std::size_t srcPos, char *dest, std::size_t destPos, std::size_t length)

  Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.
- static void **arraycopy** (**const** unsigned char *src, std::size_t srcPos, unsigned char *dest, std::size_t destPos, std::size_t length)

  Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.
- static void **arraycopy** (**const** short *src, std::size_t srcPos, short *dest, std::size_t destPos, std::size_t length)

  Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.
- static void **arraycopy** (**const** int *src, std::size_t srcPos, int *dest, std::size_t destPos, std::size_t length)

  Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.
- static void **arraycopy** (**const** long long *src, std::size_t srcPos, long long *dest, std::size_t destPos, std::size_t length)

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (**const** float *src, std::size_t srcPos, float *dest, std::size_t destPos, std::size_t length)

  Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static void **arraycopy** (**const** double *src, std::size_t srcPos, double *dest, std::size_t destPos, std::size_t length)

  Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- template<typename E >
  static void **arraycopy** (**const** E *src, std::size_t srcPos, E *dest, std::size_t destPos, std::size_t length)

  Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

- static **const util::Map**
  < std::string, std::string > & **getenv** ()

  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)

  Reads an environment value from the system and returns it as a string object.

- static void **unsetenv** (**const** std::string &name)

  Clears a set environment value if one is set.

- static void **setenv** (**const** std::string &name, **const** std::string &value)

  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.

- static **decaf::util::Properties** & **getProperties** ()

  Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.

- static std::string **getProperty** (**const** std::string &key)

  *Gets the specified **System** (p. 2065) property if set, otherwise returns an empty string.*

- static std::string **getProperty** (**const** std::string &key, **const** std::string &defaultValue)

  *Gets the specified **System** (p. 2065) property if set, otherwise returns the specified default value.*

- static std::string **setProperty** (**const** std::string &key, **const** std::string &value)

  *Sets the **System** (p. 2065) Property to the specified value.*

- static std::string **clearProperty** (**const** std::string &key)

  Clear any value associated with the system property specified.

Protected Member Functions

- **System** ()

Friends

- class **decaf::lang::Runtime**

6.533.1 Detailed Description

The **System** (p. 2065) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

Since

1.0

6.533.2 Constructor & Destructor Documentation

6.533.2.1 decaf::lang::System::System ( ) [protected]

6.533.2.2 virtual decaf::lang::System::~System ( ) [inline, virtual]

6.533.3 Member Function Documentation

6.533.3.1 static void decaf::lang::System::arraycopy (const char * *src*, std::size_t *srcPos*, char * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

Referenced by decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::add(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addAllAbsent(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::addIfAbsent(), decaf::lang::ArrayPointer< decaf::util::LinkedList< Pointer< MessageDispatch > > >::clone(), decaf::util::ArrayList< E >::ensureCapacity(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAll(), decaf::util::ArrayList< E >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::removeAt(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::retainAll(), decaf::util::concurrent::CopyOnWriteArrayList< ServiceListener * >::set(), and decaf::util::ArrayList< E >::trimToSize().

6.533.3.2 static void decaf::lang::System::arraycopy (const unsigned char * *src*, std::size_t *srcPos*, unsigned char * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

6.533.3.3 static void decaf::lang::System::arraycopy (const short * *src*, std::size_t *srcPos*, short * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

6.533.3.4 static void decaf::lang::System::arraycopy (const int * *src*, std::size_t *srcPos*, int * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

6.533.3.5 static void decaf::lang::System::arraycopy (const long long * *src*, std::size_t *srcPos*, long long * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

6.533.3.6 static void decaf::lang::System::arraycopy (const float * *src*, std::size_t *srcPos*, float * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

6.533.3.7 static void decaf::lang::System::arraycopy (const double * *src*, std::size_t *srcPos*, double * *dest*, std::size_t *destPos*, std::size_t *length*) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

6.533.3.8 template<typename E > static void decaf::lang::System::arraycopy (const E * *src*, std::size_t *srcPos*, E * *dest*, std::size_t *destPos*, std::size_t *length*) [inline, static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

Parameters

src	The source array to copy from.
srcPos	The position in the array to start copying from.
dest	The destination array to copy to.
destPos	The position in the destination array to start writing at.
length	The number of elements to copy from src to dest.

Exceptions

NullPointerException	if src or dest are NULL.

6.533.3.9 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.

6.533.3.10 static std::string decaf::lang::System::clearProperty (const std::string & *key*) [static]

Clear any value associated with the system property specified.

Parameters

key	The key name of the system property to clear.

Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

Exceptions

IllegalArgumentException	if key is an empty string.

6.533.3.11 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.

6.533.3.12 static const util::Map<std::string, std::string>& decaf::lang::System::getenv ( ) [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. 990)*	if an error occurs while getting the Environment Map.

6.533.3.13 static std::string decaf::lang::System::getenv (const std::string & *name*) [static]

Reads an environment value from the system and returns it as a string object.

Parameters

name	The environment variable to read.

Returns

a string with the value from the variables or ""

Exceptions

an	**Exception** (p. 990) if an error occurs while reading the Env.

6.533.3.14 static decaf::util::Properties& decaf::lang::System::getProperties ( ) [static]

Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Returns

a reference to the static system Properties object.

6.533.3.15 static std::string decaf::lang::System::getProperty (const std::string & *key*) [static]

Gets the specified **System** (p. 2065) property if set, otherwise returns an empty string.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Parameters

key	The key name of the desired system property to retrieve.

Returns

an empty string if the named property is not set, otherwise returns the value.

Exceptions

IllegalArgumentException	if key is an empty string.

6.533.3.16 static std::string decaf::lang::System::getProperty (const std::string & *key,* const std::string & *defaultValue*) [static]

Gets the specified **System** (p. 2065) property if set, otherwise returns the specified default value.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

Parameters

key	The key name of the desired system property to retrieve.
defaultValue	The default value to return if the key is not set in the **System** (p. 2065) properties.

Returns

the value of the named system property or the defaultValue if the property isn't set..

Exceptions

IllegalArgumentException	if key is an empty string.

6.533.3.17 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. 2071); // ... the code being measured ... long long estimatedTime = **System::nanoTime()** (p. 2071) - startTime;

Returns

The current value of the system timer, in nanoseconds.

6.533.3.18 static void decaf::lang::System::setenv (const std::string & *name,* const std::string & *value*) [static]

Sets the specified system property to the value given.

Parameters

name	The name of the environment variables to set.
value	The value to assign to name.

Exceptions

an	**Exception** (p. 990) if an error occurs when setting the environment variable.

6.533.3.19 static std::string decaf::lang::System::setProperty (const std::string & *key,* const std::string & *value*) [static]

Sets the **System** (p. 2065) Property to the specified value.

Parameters

key	The key name of the system property to set to the given value.
value	The value to assign to the key.

Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

Exceptions

IllegalArgumentException	if key is an empty string.

6.533.3.20 static void decaf::lang::System::unsetenv (const std::string & *name*) [static]

Clears a set environment value if one is set.

Parameters

name	The environment variables to clear.

Exceptions

an	**Exception** (p. 990) if an error occurs while reading the environment.

6.533.4 Friends And Related Function Documentation

6.533.4.1 friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**System.h**

6.534 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.

6.534.1 Detailed Description

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

Since

3.0

6.534.2 Constructor & Destructor Documentation

6.534.2.1 virtual activemq::threads::Task::~Task ( ) [inline, virtual]

6.534.3 Member Function Documentation

6.534.3.1 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::transport::failover::FailoverTransport** (p. 1015), **activemq::core::ActiveMQSessionExecutor** (p. 278), **activemq::threads::CompositeTaskRunner** (p. 694), **activemq::transport::failover::BackupTransportPool** (p. 380), and **activemq::transport::failover::CloseTransportsTask** (p. 635).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**Task.h**

6.535 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** (p. 2074)'s thread has exited.*
- virtual void **wakeup** ()=0

  *Signal the **TaskRunner** (p. 2074) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2073) instance will be run until its iterate method has returned false indicating it is done.*

6.535.1 Constructor & Destructor Documentation

6.535.1.1 virtual activemq::threads::TaskRunner::~TaskRunner ( ) [inline, virtual]

6.535.2 Member Function Documentation

6.535.2.1 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.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 694), and **activemq::threads::DedicatedTaskRunner** (p. 887).

6.535.2.2 virtual void activemq::threads::TaskRunner::shutdown ( ) [pure virtual]

Shutdown once the task has finished and the **TaskRunner** (p. 2074)'s thread has exited.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 694), and **activemq::threads::DedicatedTaskRunner** (p. 887).

6.535.2.3 virtual void activemq::threads::TaskRunner::wakeup ( ) [pure virtual]

Signal the **TaskRunner** (p. 2074) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2073) instance will be run until its iterate method has returned false indicating it is done.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 694), and **activemq::threads::DedicatedTaskRunner** (p. 887).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**TaskRunner.h**

6.536 decaf::internal::net::tcp::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

`#include <src/main/decaf/internal/net/tcp/TcpSocket.h>`

Inheritance diagram for decaf::internal::net::tcp::TcpSocket:

Public Member Functions

- **TcpSocket** ()

  Construct a non-connected socket.
- virtual ~**TcpSocket** ()

  Releases the socket handle but not gracefully shut down the connection.
- SocketHandle **getSocketHandle** ()

  Gets the handle for the socket.
- bool **isConnected** () **const**
- bool **isClosed** () **const**
- virtual std::string **getLocalAddress** () **const**

  *Gets the value of the local Inet address the **Socket** (p. 1900) is bound to if bound, otherwise return the **InetAddress** (p. 1113) ANY value "0.0.0.0".*

  Returns

  the local address bound to, or ANY.
- virtual void **create** ()

  *Creates the underlying platform **Socket** (p. 1900) data structures which allows for **Socket** (p. 1900) options to be applied.*
  The created socket is in an unconnected state.

Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual void **accept** (SocketImpl ∗socket)

- virtual void **bind** (**const** std::string &ipaddress, int **port**)

  *Binds this **Socket*** (p. *1900*) *instance to the local ip address and port number given.*

  Parameters

ipaddress	*The address of local ip to bind to.*
port	*The port number on the host to bind to.*

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual void **connect** (**const** std::string &hostname, int **port**, int timeout)

  Connects this socket to the given host and port.

  Parameters

hostname	*The name of the host to connect to, or IP address.*
port	*The port number on the host to connect to.*
timeout	*Time in milliseconds to wait for a connection, 0 indicates forever.*

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*
SocketTimeoutException (p. 1932)	*if the connect call times out due to timeout being set.*
IllegalArguementException	*if a parameter has an illegal value.*

- virtual void **listen** (int backlog)

  Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument. If a connection indication arrives when the queue is full, the connection is refused.

  Parameters

backlog	*The maximum length of the connection queue.*

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual **decaf::io::InputStream** ∗ **getInputStream** ()

  *Gets the InputStream linked to this **Socket*** (p. *1900*).

  Returns

  *an InputStream pointer owned by the **Socket*** (p. *1900*) *object.*

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual **decaf::io::OutputStream** ∗ **getOutputStream** ()

  *Gets the OutputStream linked to this **Socket*** (p. *1900*).

  Returns

  *an OutputStream pointer owned by the **Socket*** (p. *1900*) *object.*

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual int **available** ()

  *Gets the number of bytes that can be read from the **Socket*** (p. *1900*) *without blocking.*

  Returns

  *the number of bytes that can be read from the **Socket*** (p. *1900*) *without blocking.*

Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual void **close** ()

  Closes the socket, terminating any blocked reads or writes.

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual void **shutdownInput** ()

  Places the input stream for this socket at "end of stream".
  *Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 1926) on the socket, the stream will return EOF.*

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual void **shutdownOutput** ()

  Disables the output stream for this socket.
  For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence.
  *If you write to a socket output stream after invoking **shutdownOutput()** (p. 1927) on the socket, the stream will throw an IOException.*

  Exceptions

IOException	*if an I/O error occurs while attempting this operation.*

- virtual int **getOption** (int option) **const**

  *Gets the specified **Socket** (p. 1900) option.*

  Parameters

option	*The **Socket** (p. 1900) options whose value is to be retrieved.*

  Returns

  the value of the given socket option.

  Exceptions

IOException	*if an I/O error occurs while performing this operation.*

- virtual void **setOption** (int option, int value)

  *Sets the specified option on the **Socket** (p. 1900) if supported.*

  Parameters

option	*The **Socket** (p. 1900) option to set.*
value	*The value of the socket option to apply to the socket.*

  Exceptions

IOException	*if an I/O error occurs while performing this operation.*

- int **read** (unsigned char *buffer, int size, int offset, int length)

  Reads the requested data from the Socket and write it into the passed in buffer.
- void **write** (**const** unsigned char *buffer, int size, int offset, int length)

  Writes the specified data in the passed in buffer to the Socket.

Protected Member Functions

- void **checkResult** (apr_status_t value) **const**

6.536.1 Detailed Description

Platform-independent implementation of the socket interface.

6.536.2 Constructor & Destructor Documentation

6.536.2.1 decaf::internal::net::tcp::TcpSocket::TcpSocket ( )

Construct a non-connected socket.

Exceptions

SocketException	thrown if an error occurs while creating the Socket.

6.536.2.2 virtual decaf::internal::net::tcp::TcpSocket::~TcpSocket ( ) [virtual]

Releases the socket handle but not gracefully shut down the connection.

6.536.3 Member Function Documentation

6.536.3.1 virtual void decaf::internal::net::tcp::TcpSocket::accept (SocketImpl * *socket*) [virtual]

6.536.3.2 virtual int decaf::internal::net::tcp::TcpSocket::available ( ) [virtual]

Gets the number of bytes that can be read from the **Socket** (p. 1900) without blocking.

Returns

the number of bytes that can be read from the **Socket** (p. 1900) without blocking.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1923).

6.536.3.3 virtual void decaf::internal::net::tcp::TcpSocket::bind (const std::string & *ipaddress*, int *port*) [virtual]

Binds this **Socket** (p. 1900) instance to the local ip address and port number given.

Parameters

ipaddress	The address of local ip to bind to.
port	The port number on the host to bind to.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1923).

6.536.3.4 void decaf::internal::net::tcp::TcpSocket::checkResult (apr_status_t *value*) const [protected]

6.536.3.5 virtual void decaf::internal::net::tcp::TcpSocket::close ( ) [virtual]

Closes the socket, terminating any blocked reads or writes.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1923).

6.536.3.6 virtual void decaf::internal::net::tcp::TcpSocket::connect (const std::string & *hostname,* int *port,* int *timeout*) [virtual]

Connects this socket to the given host and port.

Parameters

hostname	The name of the host to connect to, or IP address.
port	The port number on the host to connect to.
timeout	Time in milliseconds to wait for a connection, 0 indicates forever.

Exceptions

IOException	if an I/O error occurs while attempting this operation.
SocketTimeoutException *(p. 1932)*	if the connect call times out due to timeout being set.
IllegalArguementException	if a parameter has an illegal value.

Implements **decaf::net::SocketImpl** (p. 1923).

6.536.3.7 virtual void decaf::internal::net::tcp::TcpSocket::create ( ) [virtual]

Creates the underlying platform **Socket** (p. 1900) data structures which allows for **Socket** (p. 1900) options to be applied.

The created socket is in an unconnected state.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1924).

6.536.3.8 virtual decaf::io::InputStream* decaf::internal::net::tcp::TcpSocket::getInputStream ( ) [virtual]

Gets the InputStream linked to this **Socket** (p. 1900).

Returns

an InputStream pointer owned by the **Socket** (p. 1900) object.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1924).

6.536.3.9 virtual std::string decaf::internal::net::tcp::TcpSocket::getLocalAddress ( ) const [virtual]

Gets the value of the local Inet address the **Socket** (p. 1900) is bound to if bound, otherwise return the **InetAddress** (p. 1113) ANY value "0.0.0.0".

Returns

the local address bound to, or ANY.

Implements **decaf::net::SocketImpl** (p. 1925).

6.536.3.10 virtual int decaf::internal::net::tcp::TcpSocket::getOption (int *option*) const [virtual]

Gets the specified **Socket** (p. 1900) option.

Parameters

option	The **Socket** (p. 1900) options whose value is to be retrieved.

Returns

the value of the given socket option.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Implements **decaf::net::SocketImpl** (p. 1925).

6.536.3.11 virtual decaf::io::OutputStream* decaf::internal::net::tcp::TcpSocket::getOutputStream ( ) [virtual]

Gets the OutputStream linked to this **Socket** (p. 1900).

Returns

an OutputStream pointer owned by the **Socket** (p. 1900) object.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1925).

6.536.3.12 SocketHandle decaf::internal::net::tcp::TcpSocket::getSocketHandle ( ) [inline]

Gets the handle for the socket.

Returns

SocketHabler for this Socket, can be NULL

6.536.3.13 bool decaf::internal::net::tcp::TcpSocket::isClosed ( ) const [inline]

Returns

true if the close method has been called on this Socket.

6.536.3.14 bool decaf::internal::net::tcp::TcpSocket::isConnected ( ) const [inline]

Returns

true if the socketHandle is not in a disconnected state.

6.536.3.15 virtual void decaf::internal::net::tcp::TcpSocket::listen (int *backlog*) [virtual]

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

Parameters

backlog	The maximum length of the connection queue.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1926).

6.536.3.16 int decaf::internal::net::tcp::TcpSocket::read (unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Reads the requested data from the Socket and write it into the passed in buffer.

Parameters

buffer	The buffer to read into
size	The size of the specified buffer
offset	The offset into the buffer where reading should start filling.
length	The number of bytes past offset to fill with data.

Returns

the actual number of bytes read or -1 if at EOF.

Exceptions

IOException	if an I/O error occurs during the read.
NullPointerException	if buffer is Null.
IndexOutOfBounds-Exception	if offset + length is greater than buffer size.

6.536.3.17 virtual void decaf::internal::net::tcp::TcpSocket::setOption (int *option*, int *value*) [virtual]

Sets the specified option on the **Socket** (p. 1900) if supported.

Parameters

option	The **Socket** (p. 1900) option to set.
value	The value of the socket option to apply to the socket.

Exceptions

IOException	if an I/O error occurs while performing this operation.

Implements **decaf::net::SocketImpl** (p. 1926).

6.536.3.18 virtual void decaf::internal::net::tcp::TcpSocket::shutdownInput ( ) [virtual]

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 1926) on the socket, the stream will return EOF.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1926).

6.536.3.19 virtual void decaf::internal::net::tcp::TcpSocket::shutdownOutput ( ) [virtual]

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 1927) on the socket, the stream will throw an IOException.

Exceptions

IOException	if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 1927).

6.536.3.20 void decaf::internal::net::tcp::TcpSocket::write (const unsigned char * *buffer*, int *size*, int *offset*, int *length*)

Writes the specified data in the passed in buffer to the Socket.

Parameters

buffer	The buffer to write to the socket.
size	The size of the specified buffer.
offset	The offset into the buffer where the data to write starts at.
length	The number of bytes past offset to write.

Exceptions

IOException	if an I/O error occurs during the write.
NullPointerException	if buffer is Null.
IndexOutOfBounds-Exception	if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/**TcpSocket.h**

6.537 decaf::internal::net::tcp::TcpSocketInputStream Class Reference

Input stream for performing reads on a socket.

`#include <src/main/decaf/internal/net/tcp/TcpSocketInputStream.h>`

Inheritance diagram for decaf::internal::net::tcp::TcpSocketInputStream:

Public Member Functions

- **TcpSocketInputStream** (**TcpSocket** ∗socket)

  Create a new InputStream to use for reading from the TCP/IP socket.
- virtual ∼**TcpSocketInputStream** ()
- virtual int **available** () **const**

  Indicates the number of bytes available.
  The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.
  The default implementation of this method returns zero.

  Returns

  the number of bytes available on this input stream.

  Exceptions

IOException (p. 1198)	*if an I/O error occurs.*

- virtual void **close** ()

  Close - does nothing.
- virtual long long **skip** (long long num)

  Not supported.

Protected Member Functions

- virtual int **doReadByte** ()
- virtual int **doReadArrayBounded** (unsigned char ∗buffer, int size, int offset, int length)

6.537.1 Detailed Description

Input stream for performing reads on a socket.

This class will only work properly for blocking sockets.

Since

1.0

6.537.2 Constructor & Destructor Documentation

6.537.2.1 decaf::internal::net::tcp::TcpSocketInputStream::TcpSocketInputStream (TcpSocket * *socket*)

Create a new InputStream to use for reading from the TCP/IP socket.

Parameters

socket	The parent SocketImpl for this stream.

6.537.2.2 virtual decaf::internal::net::tcp::TcpSocketInputStream::~TcpSocketInputStream ( ) [virtual]

6.537.3 Member Function Documentation

6.537.3.1 virtual int decaf::internal::net::tcp::TcpSocketInputStream::available ( ) const [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

Returns

the number of bytes available on this input stream.

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.537.3.2 virtual void decaf::internal::net::tcp::TcpSocketInputStream::close ( ) [virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 1134) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Exceptions

IOException *(*p. *1198)*	if an I/O error occurs while closing the **InputStream** (p. 1134).

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.537.3.3 virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadArrayBounded (unsigned char * *buffer*, int *size*, int *offset*, int *length*) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 1136).

6.537.3.4 virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadByte ( ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 1137).

6.537.3.5 virtual long long decaf::internal::net::tcp::TcpSocketInputStream::skip (long long *num*) [virtual]

Not supported.

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.

The skip method of **InputStream** (p. 1134) creates a byte array and then repeatedly reads into it until num 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. 1198)*	if an I/O error occurs.
UnsupportedOperation-Exception	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 1140).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/**TcpSocketInputStream.h**

6.538 decaf::internal::net::tcp::TcpSocketOutputStream Class Reference

Output stream for performing write operations on a socket.

`#include <src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h>`

Inheritance diagram for decaf::internal::net::tcp::TcpSocketOutputStream:

Public Member Functions

- **TcpSocketOutputStream** (**TcpSocket** ∗socket)

  Create a new instance of a Socket OutputStream class.
- virtual ∼**TcpSocketOutputStream** ()
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
  The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1198)	*if an error occurs while closing.*

The default implementation of this method does nothing.

Protected Member Functions

- virtual void **doWriteByte** (unsigned char c)
- virtual void **doWriteArrayBounded** (**const** unsigned char ∗buffer, int size, int offset, int length)

6.538.1 Detailed Description

Output stream for performing write operations on a socket.

Since

1.0

6.538.2 Constructor & Destructor Documentation

6.538.2.1 decaf::internal::net::tcp::TcpSocketOutputStream::TcpSocketOutputStream (TcpSocket ∗ *socket*)

Create a new instance of a Socket OutputStream class.

Parameters

socket	The socket to use to write out the data.

6.538.2.2 virtual decaf::internal::net::tcp::TcpSocketOutputStream::∼TcpSocketOutputStream ( ) [virtual]

6.538.3 Member Function Documentation

6.538.3.1 virtual void decaf::internal::net::tcp::TcpSocketOutputStream::close ( ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException *(*p. *1198)*	if an error occurs while closing.

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.538.3.2 virtual void decaf::internal::net::tcp::TcpSocketOutputStream::doWriteArrayBounded (const unsigned char ∗ *buffer,* int *size,* int *offset,* int *length*) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 1602).

6.538.3.3 virtual void decaf::internal::net::tcp::TcpSocketOutputStream::doWriteByte (unsigned char *c*) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 1602).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/**TcpSocketOutputStream.h**

6.539 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. 1200).

`#include <src/main/activemq/transport/tcp/TcpTransport.h>`

Inheritance diagram for activemq::transport::tcp::TcpTransport:

Public Member Functions

- **TcpTransport** (**const Pointer**< **Transport** > &**next**)

  *Creates a new instance of a **TcpTransport** (p. 2086), the transport is left unconnected and is in a unusable state until the connect method is called.*
- virtual ~**TcpTransport** ()
- void **connect** (**const decaf::net::URI** &uri, **const decaf::util::Properties** &properties)

  Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.
- virtual void **close** ()

  Closes this object and deallocates the appropriate resources.
- virtual bool **isFaultTolerant** () **const**

  *Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () **const**

  *Is the **Transport** (p. 2161) Connected to its Broker.*
- virtual bool **isClosed** () **const**

  *Has the **Transport** (p. 2161) been shutdown and no longer usable.*

Protected Member Functions

- virtual **decaf::net::Socket** * **createSocket** ()

  Create an unconnected Socket instance to be used by the transport to communicate with the broker.
- virtual void **configureSocket** (**decaf::net::Socket** *socket, **const decaf::util::Properties** &properties)

  Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

6.539.1 Detailed Description

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1200).

The lower level transport should take care of managing stream reads and writes.

6.539.2 Constructor & Destructor Documentation

6.539.2.1 activemq::transport::tcp::TcpTransport::TcpTransport (const Pointer< Transport > & *next*)

Creates a new instance of a **TcpTransport** (p. 2086), the transport is left unconnected and is in a unusable state until the connect method is called.

Parameters

next	The next transport in the chain

6.539.2.2 virtual activemq::transport::tcp::TcpTransport::~TcpTransport ( ) [virtual]

6.539.3 Member Function Documentation

6.539.3.1 virtual void activemq::transport::tcp::TcpTransport::close ( ) [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.

Reimplemented from **activemq::transport::TransportFilter** (p. 2170).

6.539.3.2 virtual void activemq::transport::tcp::TcpTransport::configureSocket (decaf::net::Socket * *socket*, const decaf::util::Properties & *properties*) [protected, virtual]

Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

Subclasses can override this option to set more configuration options, they should called the base class version to allow the default set of Socket options to also be configured.

Parameters

socket	The Socket instance to configure using options from the given Properties.

Exceptions

NullPointerException	if the Socket instance is null.
IllegalArgumentException	if the socket instance is not handled by the class.
SocketException	if there is an error while setting one of the Socket options.

6.539.3.3 void activemq::transport::tcp::TcpTransport::connect (const decaf::net::URI & *uri*, const decaf::util::Properties & *properties*)

Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.

The Socket is configured using parameters in the properties that are passed to this method.

Parameters

uri	The URI that the **Transport** (p. 2161) is to connect to once initialized.
properties	The Properties that have been parsed from the URI or from configuration files.

6.539.3.4 virtual decaf::net::Socket* activemq::transport::tcp::TcpTransport::createSocket ( ) [protected, virtual]

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

Returns

a newly created unconnected Socket instance.

Exceptions

IOException	if there is an error while creating the unconnected Socket.

Reimplemented in **activemq::transport::tcp::SslTransport** (p. 1957).

6.539.3.5 virtual bool activemq::transport::tcp::TcpTransport::isClosed ( ) const [inline, virtual]

Has the **Transport** (p. 2161) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2161)

Reimplemented from **activemq::transport::TransportFilter** (p. 2172).

6.539.3.6 virtual bool activemq::transport::tcp::TcpTransport::isConnected ( ) const [inline, virtual]

Is the **Transport** (p. 2161) Connected to its Broker.

Returns

true if a connection has been made.

Reimplemented from **activemq::transport::TransportFilter** (p. 2172).

6.539.3.7 virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant ( ) const [inline, virtual]

Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2161) is fault tolerant.

Reimplemented from **activemq::transport::TransportFilter** (p. 2172).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransport.h**

6.540 activemq::transport::tcp::TcpTransportFactory Class Reference

Factory Responsible for creating the **TcpTransport** (p. 2086).

`#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)

  *Creates a fully configured **Transport** (p. 2161) instance which could be a chain of filters and transports.*
- virtual **Pointer**< **Transport** > **createComposite** (**const decaf::net::URI** &location)

  *Creates a slimed down **Transport** (p. 2161) 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)

6.540.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 2086).

6.540.2 Constructor & Destructor Documentation

6.540.2.1 virtual activemq::transport::tcp::TcpTransportFactory::~TcpTransportFactory ( ) [inline, virtual]

6.540.3 Member Function Documentation

6.540.3.1 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::create (const decaf::net::URI & *location*) [virtual]

Creates a fully configured **Transport** (p. 2161) 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. 2168).

6.540.3.2 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::createComposite (const decaf::net::URI & *location*) [virtual]

Creates a slimed down **Transport** (p. 2161) 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. 2168).

6.540.3.3 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::doCreateComposite (const decaf::net::URI & *location,* const Pointer< wireformat::WireFormat > & *wireFormat,* const decaf::util::Properties & *properties*) [protected, virtual]

Reimplemented in **activemq::transport::tcp::SslTransportFactory** (p. 1958).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransportFactory.h**

6.541 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 1722) based **Destination** (p. 936).

`#include <src/main/cms/TemporaryQueue.h>`

Inheritance diagram for cms::TemporaryQueue:

Public Member Functions

- virtual ~**TemporaryQueue** () throw ()
- virtual std::string **getQueueName** () **const** =0

  Gets the name of this queue.
- virtual void **destroy** ()=0

  *Destroy's the Temporary **Destination** (p. 936) at the Provider.*

6.541.1 Detailed Description

Defines a Temporary **Queue** (p. 1722) based **Destination** (p. 936).

A **TemporaryQueue** (p. 2090) is a special type of **Queue** (p. 1722) **Destination** (p. 936) that can only be consumed from the **Connection** (p. 725) which created it. TemporaryQueues are most commonly used as the reply to address for **Message** (p. 1426)'s that implement the request response pattern.

A **TemporaryQueue** (p. 2090) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 725) that created it.

Since

1.0

6.541.2 Constructor & Destructor Documentation

6.541.2.1 virtual cms::TemporaryQueue::~TemporaryQueue ( ) throw () [virtual]

6.541.3 Member Function Documentation

6.541.3.1 virtual void cms::TemporaryQueue::destroy ( ) [pure virtual]

Destroy's the Temporary **Destination** (p. 936) at the Provider.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 301).

6.541.3.2 virtual std::string cms::TemporaryQueue::getQueueName ( ) const [pure virtual]

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 303).

The documentation for this class was generated from the following file:

- src/main/cms/**TemporaryQueue.h**

6.542 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 2141) based **Destination** (p. 936).

`#include <src/main/cms/TemporaryTopic.h>`

Inheritance diagram for cms::TemporaryTopic:

Public Member Functions

- virtual ~**TemporaryTopic** () throw ()
- virtual std::string **getTopicName** () **const** =0

  Gets the name of this topic.
- virtual void **destroy** ()=0

  Destroy's the Temporary ***Destination*** *(p. 936) at the Provider.*

6.542.1 Detailed Description

Defines a Temporary **Topic** (p. 2141) based **Destination** (p. 936).

A **TemporaryTopic** (p. 2091) is a special type of **Topic** (p. 2141) **Destination** (p. 936) that can only be consumed from the **Connection** (p. 725) which created it. TemporaryTopics are most commonly used as the reply to address for **Message** (p. 1426)'s that implement the request response pattern.

A **TemporaryTopic** (p. 2091) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 725) that created it.

Since

1.0

6.542.2 Constructor & Destructor Documentation

6.542.2.1 virtual cms::TemporaryTopic::~TemporaryTopic ( ) throw () [virtual]

6.542.3 Member Function Documentation

6.542.3.1 virtual void cms::TemporaryTopic::destroy ( ) [pure virtual]

Destroy's the Temporary **Destination** (p. 936) at the Provider.

Exceptions

CMSException *(p. 640)*	

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 308).

6.542.3.2 virtual std::string cms::TemporaryTopic::getTopicName ( ) const [pure virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 310).

The documentation for this class was generated from the following file:

- src/main/cms/**TemporaryTopic.h**

6.543 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** () throw ()
- virtual std::string **getText** () **const** =0

  Gets the message character buffer.
- virtual void **setText** (**const** char *msg)=0

  Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void **setText** (**const** std::string &msg)=0

  Sets the message contents.

6.543.1 Detailed Description

Interface for a text message.

A **TextMessage** (p. 2093) can contain any Text based pay load such as an XML Document or other Text based document.

Like all Messages, a **TextMessage** (p. 2093) is received in Read-Only mode, any attempt to write to the **Message** (p. 1426) 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

6.543.2 Constructor & Destructor Documentation

6.543.2.1 virtual cms::TextMessage::~TextMessage ( ) throw () [virtual]

6.543.3 Member Function Documentation

6.543.3.1 virtual std::string cms::TextMessage::getText ( ) const [pure virtual]

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 316).

6.543.3.2 virtual void cms::TextMessage::setText (const char * *msg*) [pure virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

msg	The message buffer.

Exceptions

CMSException *(p. 640)*	- if an internal error occurs.
MessageNotWriteable-Exception *(p. 1490)*	- if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 317).

6.543.3.3 virtual void cms::TextMessage::setText (const std::string & *msg*) [pure virtual]

Sets the message contents.

Parameters

msg	The message buffer.

Exceptions

CMSException *(*p. *640)*	- if an internal error occurs.
MessageNotWriteable-Exception *(*p. *1490)*	- if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 317).

The documentation for this class was generated from the following file:

- src/main/cms/**TextMessage.h**

6.544 decaf::lang::Thread Class Reference

A **Thread** (p. 2094) is a concurrent unit of execution.

`#include <src/main/decaf/lang/Thread.h>`

Inheritance diagram for decaf::lang::Thread:

Data Structures

- class **UncaughtExceptionHandler**

  Interface for handlers invoked when a ***Thread*** *(*p. *2094) abruptly terminates due to an uncaught exception.*

Public Types

- enum **State** {
  NEW = 0, **RUNNABLE** = 1, **BLOCKED** = 2, **WAITING** = 3,
  TIMED_WAITING = 4, **SLEEPING** = 5, **TERMINATED** = 6 }

  Represents the various states that the ***Thread*** *(*p. *2094) can be in during its lifetime.*

Public Member Functions

- **Thread** ()

  Constructs a new ***Thread*** *(*p. *2094).*
- **Thread** (**Runnable** ∗task)

  Constructs a new ***Thread*** *(*p. *2094) with the given target* ***Runnable*** *(*p. *1792) task.*
- **Thread** (**const** std::string &name)

  Constructs a new ***Thread*** *(*p. *2094) with the given name.*
- **Thread** (**Runnable** ∗task, **const** std::string &name)

  Constructs a new ***Thread*** *(*p. *2094) with the given target* ***Runnable*** *(*p. *1792) task and name.*
- virtual ∼**Thread** ()
- virtual void **start** ()

  Creates a system thread and starts it in a joinable mode.
- virtual void **join** ()

*Forces the Current **Thread** (p. 2094) to wait until the thread exits.*

- virtual void **join** (long long millisecs)

  *Forces the Current **Thread** (p. 2094) to wait until the thread exits.*
- virtual void **join** (long long millisecs, unsigned int nanos)

  *Forces the Current **Thread** (p. 2094) to wait until the thread exits.*
- virtual void **run** ()

  Default implementation of the run method - does nothing.
- std::string **getName** () **const**

  *Returns the **Thread** (p. 2094)'s assigned name.*
- void **setName** (**const** std::string &name)

  *Sets the name of the **Thread** (p. 2094) to the new Name given by the argument name*
- int **getPriority** () **const**

  *Gets the currently set priority for this **Thread** (p. 2094).*
- void **setPriority** (int value)

  *Sets the current **Thread** (p. 2094)'s priority to the newly specified value.*
- void **setDaemon** (bool value)

  *Sets if the given **Thread** (p. 2094) is a Daemon **Thread** (p. 2094) or not.*
- bool **isDaemon** () **const**

  Returns whether this thread is a daemon thread or not, if true this thread cannot be joined.
- **const UncaughtExceptionHandler** * **getUncaughtExceptionHandler** () **const**

  Set the handler invoked when this thread abruptly terminates due to an uncaught exception.
- void **setUncaughtExceptionHandler** (**UncaughtExceptionHandler** *handler)

  Set the handler invoked when this thread abruptly terminates due to an uncaught exception.
- std::string **toString** () **const**

  *Returns a string that describes the **Thread** (p. 2094).*
- bool **isAlive** () **const**

  *Returns true if the **Thread** (p. 2094) is alive, meaning it has been started and has not yet died.*
- **Thread::State getState** () **const**

  *Returns the currently set State of this **Thread** (p. 2094).*

Static Public Member Functions

- static void **sleep** (long long millisecs)

  Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.
- static void **sleep** (long long millisecs, unsigned int nanos)

  Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.
- static void **yield** ()

  Causes the currently executing thread object to temporarily pause and allow other threads to execute.
- static long long **getId** ()

  *Obtains the **Thread** (p. 2094) Id of the current thread.*
- static **Thread** * **currentThread** ()

  Returns a pointer to the currently executing thread object.

Static Public Attributes

- static **const** int **MIN_PRIORITY** = 1

  The minimum priority that a thread can have.
- static **const** int **NORM_PRIORITY** = 5

  The default priority that a thread is given at create time.
- static **const** int **MAX_PRIORITY** = 10

  The maximum priority that a thread can have.

Friends

- class **decaf::util::concurrent::locks::LockSupport**
- class **decaf::lang::Runtime**

6.544.1 Detailed Description

A **Thread** (p. 2094) is a concurrent unit of execution.

It has its own call stack for methods being invoked, their arguments and local variables. Each process has at least one main **Thread** (p. 2094) running when it is started; typically, there are several others for housekeeping. The application might decide to launch additional Threads for specific purposes.

Threads in the same process interact and synchronize by the use of shared objects and monitors associated with these objects.

There are basically two main ways of having a **Thread** (p. 2094) execute application code. One is providing a new class that extends **Thread** (p. 2094) and overriding its **run()** (p. 2100) method. The other is providing a new **Thread** (p. 2094) instance with a **Runnable** (p. 1792) object during its creation. In both cases, the **start()** (p. 2101) method must be called to actually execute the new **Thread** (p. 2094).

Each **Thread** (p. 2094) has an integer priority that basically determines the amount of CPU time the **Thread** (p. 2094) gets. It can be set using the **setPriority(int)** (p. 2100) method. A **Thread** (p. 2094) can also be made a daemon, which makes it run in the background. The latter also affects VM termination behavior: the VM does not terminate automatically as long as there are non-daemon threads running.

See also

decaf.lang.ThreadGroup (p. 2103)

Since

1.0

6.544.2 Member Enumeration Documentation

6.544.2.1 enum decaf::lang::Thread::State

Represents the various states that the **Thread** (p. 2094) can be in during its lifetime.

Enumerator:

NEW Before a **Thread** (p. 2094) is started it exists in this State.

RUNNABLE While a **Thread** (p. 2094) is running and is not blocked it is in this State.

BLOCKED A **Thread** (p. 2094) that is waiting to acquire a lock is in this state.

WAITING A **Thread** (p. 2094) that is waiting for another **Thread** (p. 2094) to perform an action is in this state.

TIMED_WAITING A **Thread** (p. 2094) that is waiting for another **Thread** (p. 2094) to perform an action up to a specified time interval is in this state.

SLEEPING A **Thread** (p. 2094) that is blocked in a Sleep call is in this state.

TERMINATED A **Thread** (p. 2094) whose run method has exited is in this state.

6.544.3 Constructor & Destructor Documentation

6.544.3.1 decaf::lang::Thread::Thread ( )

Constructs a new **Thread** (p. 2094).

This constructor has the same effect as Thread(NULL, NULL, GIVEN_NAME), where GIVEN_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

6.544.3.2 decaf::lang::Thread::Thread (Runnable * *task*)

Constructs a new **Thread** (p. 2094) with the given target **Runnable** (p. 1792) task.

This constructor has the same effect as Thread(NULL, task, GIVEN_NAME), where GIVEN_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

task	the **Runnable** (p. 1792) that this thread manages, if the task is NULL the **Thread** (p. 2094)'s run method is used instead.

6.544.3.3 decaf::lang::Thread::Thread (const std::string & *name*)

Constructs a new **Thread** (p. 2094) with the given name.

This constructor has the same effect as Thread(NULL, NULL, GIVEN_NAME), where GIVEN_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

name	the name to assign to this **Thread** (p. 2094).

6.544.3.4 decaf::lang::Thread::Thread (Runnable * *task,* const std::string & *name*)

Constructs a new **Thread** (p. 2094) with the given target **Runnable** (p. 1792) task and name.

This constructor has the same effect as Thread(NULL, task, GIVEN_NAME), where GIVEN_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters

task	the **Runnable** (p. 1792) that this thread manages, if the task is NULL the **Thread** (p. 2094)'s run method is used instead.
name	the name to assign to this **Thread** (p. 2094).

6.544.3.5 virtual decaf::lang::Thread::~Thread ( ) [virtual]

6.544.4 Member Function Documentation

6.544.4.1 static Thread* decaf::lang::Thread::currentThread ( ) [static]

Returns a pointer to the currently executing thread object.

Returns

Pointer (p. 1614) to the **Thread** (p. 2094) object representing the currently running **Thread** (p. 2094).

6.544.4.2 static long long decaf::lang::Thread::getId ( ) [static]

Obtains the **Thread** (p. 2094) Id of the current thread.

Returns

Thread (p. 2094) Id

6.544.4.3 std::string decaf::lang::Thread::getName ( ) const

Returns the **Thread** (p. 2094)'s assigned name.

Returns

the Name of the **Thread** (p. 2094).

6.544.4.4 int decaf::lang::Thread::getPriority ( ) const

Gets the currently set priority for this **Thread** (p. 2094).

Returns

an int value representing the **Thread** (p. 2094)'s current priority.

6.544.4.5 Thread::State decaf::lang::Thread::getState ( ) const

Returns the currently set State of this **Thread** (p. 2094).

Returns

the **Thread** (p. 2094)'s current state.

6.544.4.6 const UncaughtExceptionHandler* decaf::lang::Thread::getUncaughtExceptionHandler ( ) const

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

Returns

a pointer to the set **UncaughtExceptionHandler** (p. 2180).

6.544.4.7 bool decaf::lang::Thread::isAlive ( ) const

Returns true if the **Thread** (p. 2094) is alive, meaning it has been started and has not yet died.

Returns

true if the thread is alive.

6.544.4.8 bool decaf::lang::Thread::isDaemon ( ) const

Returns whether this thread is a daemon thread or not, if true this thread cannot be joined.

Returns

true if the thread is a daemon thread.

6.544.4.9 virtual void decaf::lang::Thread::join ( ) [virtual]

Forces the Current **Thread** (p. 2094) to wait until the thread exits.

Exceptions

InterruptedException	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.544.4.10 virtual void decaf::lang::Thread::join (long long *millisecs*) [virtual]

Forces the Current **Thread** (p. 2094) to wait until the thread exits.

Parameters

millisecs	the time in Milliseconds before the thread resumes

Exceptions

IllegalArgumentException	if the milliseconds parameter is negative.
InterruptedException	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.544.4.11 virtual void decaf::lang::Thread::join (long long *millisecs,* unsigned int *nanos*) [virtual]

Forces the Current **Thread** (p. 2094) to wait until the thread exits.

Parameters

millisecs	the time in Milliseconds before the thread resumes
nanos	0-999999 extra nanoseconds to sleep.

Exceptions

IllegalArgumentException	if the nanoseconds parameter is out of range or the milliseconds paramter is negative.
InterruptedException	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.544.4.12 virtual void decaf::lang::Thread::run ( ) [virtual]

Default implementation of the run method - does nothing.

Implements **decaf::lang::Runnable** (p. 1793).

Reimplemented in **activemq::transport::mock::InternalCommandListener** (p. 1184).

6.544.4.13 void decaf::lang::Thread::setDaemon (bool *value*)

Sets if the given **Thread** (p. 2094) is a Daemon **Thread** (p. 2094) or not.

Daemon threads cannot be joined and its resource are automatically reclaimed when it terminates.

Parameters

value	**Boolean** (p. 422) indicating if this thread should be a daemon thread or not.

Exceptions

IllegalThreadStateException	if the thread is already active.

6.544.4.14 void decaf::lang::Thread::setName (const std::string & *name*)

Sets the name of the **Thread** (p. 2094) to the new Name given by the argument *name*

name the new name of the **Thread** (p. 2094).

6.544.4.15 void decaf::lang::Thread::setPriority (int *value*)

Sets the current **Thread** (p. 2094)'s priority to the newly specified value.

The given value must be within the rane **Thread::MIN_PRIORITY** (p. 2102) and **Thread::MAX_PRIORITY** (p. 2102).

Parameters

value	the new priority value to assign to this **Thread** (p. 2094).

Exceptions

IllegalArgumentException	if the value is out of range.

6.544.4.16 void decaf::lang::Thread::setUncaughtExceptionHandler (UncaughtExceptionHandler * *handler*)

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

Parameters

handler	the UncaightExceptionHandler to invoke when the **Thread** (p. 2094) terminates due to an uncaught exception.

6.544.4.17 static void decaf::lang::Thread::sleep (long long *millisecs*) [static]

Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

Parameters

millisecs	time in milliseconds to halt execution.

Exceptions

IllegalArgumentException	if the milliseconds parameter is negative.
InterruptedException	if the **Thread** (p. 2094) was interrupted while sleeping.

6.544.4.18 static void decaf::lang::Thread::sleep (long long *millisecs*, unsigned int *nanos*) [static]

Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

Parameters

millisecs	time in milliseconds to halt execution.
nanos	0-999999 extra nanoseconds to sleep.

Exceptions

IllegalArgumentException	if the nanoseconds parameter is out of range or the milliseconds paramter is negative.
InterruptedException	if the **Thread** (p. 2094) was interrupted while sleeping.

6.544.4.19 virtual void decaf::lang::Thread::start ( ) [virtual]

Creates a system thread and starts it in a joinable mode.

Upon creation, the **run()** (p. 2100) method of either this object or the provided **Runnable** (p. 1792) object will be invoked in the context of this thread.

Exceptions

IllegalThreadStateException	if the thread has already been started.
RuntimeException	if the **Thread** (p. 2094) cannot be created for some reason.

6.544.4.20 std::string decaf::lang::Thread::toString ( ) const

Returns a string that describes the **Thread** (p. 2094).

Returns

string describing the **Thread** (p. 2094).

6.544.4.21 static void decaf::lang::Thread::yield ( ) [static]

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

6.544.5 Friends And Related Function Documentation

6.544.5.1 friend class decaf::lang::Runtime [friend]

6.544.5.2 friend class decaf::util::concurrent::locks::LockSupport [friend]

6.544.6 Field Documentation

6.544.6.1 const int decaf::lang::Thread::MAX_PRIORITY = 10 [static]

The maximum priority that a thread can have.

6.544.6.2 const int decaf::lang::Thread::MIN_PRIORITY = 1 [static]

The minimum priority that a thread can have.

6.544.6.3 const int decaf::lang::Thread::NORM_PRIORITY = 5 [static]

The default priority that a thread is given at create time.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Thread.h**

6.545 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p. 2102)

`#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.

6.545.1 Detailed Description

public interface **ThreadFactory** (p. 2102)

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. 2102) { 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

6.545.2 Constructor & Destructor Documentation

6.545.2.1 virtual decaf::util::concurrent::ThreadFactory::∼ThreadFactory ( ) [inline, virtual]

6.545.3 Member Function Documentation

6.545.3.1 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**

6.546 decaf::lang::ThreadGroup Class Reference

`#include <src/main/decaf/lang/ThreadGroup.h>`

Public Member Functions

- **ThreadGroup** ()
- virtual ~**ThreadGroup** ()

6.546.1 Detailed Description

Since

1.0

6.546.2 Constructor & Destructor Documentation

6.546.2.1 decaf::lang::ThreadGroup::ThreadGroup ( )

6.546.2.2 virtual decaf::lang::ThreadGroup::~ThreadGroup ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**ThreadGroup.h**

6.547 decaf::util::concurrent::ThreadPoolExecutor Class Reference

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

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

Inheritance diagram for decaf::util::concurrent::ThreadPoolExecutor:

Data Structures

- class **AbortPolicy**

  *Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class always throws a **RejectedExecutionException** (p. 1755).*
- class **CallerRunsPolicy**

*Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute** (p. 2110) this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.*

Public Member Functions

- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, **const TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue)

  *Creates a new instance of a **ThreadPoolExecutor** (p. 2104).*
- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, **const TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue, **RejectedExecutionHandler** *handler)

  *Creates a new instance of a **ThreadPoolExecutor** (p. 2104).*
- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, **const TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue, **ThreadFactory** *threadFactory)

  *Creates a new instance of a **ThreadPoolExecutor** (p. 2104).*
- **ThreadPoolExecutor** (int corePoolSize, int maxPoolSize, long long keepAliveTime, **const TimeUnit** &unit, **BlockingQueue**< **decaf::lang::Runnable** * > *workQueue, **ThreadFactory** *threadFactory, **RejectedExecutionHandler** *handler)

  *Creates a new instance of a **ThreadPoolExecutor** (p. 2104).*
- virtual ~**ThreadPoolExecutor** ()
- virtual void **execute** (**decaf::lang::Runnable** *task)

  Executes the given command at some time in the future.
- virtual void **shutdown** ()

  *Performs an orderly shutdown of this **Executor** (p. 1004).*
- virtual **ArrayList**
  < **decaf::lang::Runnable** * > **shutdownNow** ()

  *Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 345) containing the Runnables that did not get executed, these object become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.*
- virtual bool **awaitTermination** (long long timeout, **const decaf::util::concurrent::TimeUnit** &unit)

  The caller will block until the executor has completed termination meaning all tasks that where scheduled before shutdown have now completed and the executor is ready for deletion.
- virtual bool **isShutdown** () **const**

  Returns whether this executor has been shutdown or not.
- virtual bool **isTerminated** () **const**

  Returns whether all tasks have completed after this executor was shut down.
- virtual int **getPoolSize** () **const**

  *Returns the number of threads that currently exists for this **Executor** (p. 1004).*
- virtual int **getCorePoolSize** () **const**

  *Returns the configured number of core threads for this **Executor** (p. 1004).*
- virtual void **setCorePoolSize** (int poolSize)

  ***Set** (p. 1857) the number of threads that this executor treats as its core threads, this value will override the value set in the constructor.*
- virtual int **getMaximumPoolSize** () **const**

  *Returns the configured maximum number of threads for this **Executor** (p. 1004).*
- virtual void **setMaximumPoolSize** (int maxSize)

  *Sets the maximum number of workers this **Executor** (p. 1004) is allowed to have at any given time above the core pool size.*
- virtual long long **getTaskCount** () **const**

  Returns the current number of pending tasks in the work queue.
- virtual int **getActiveCount** () **const**

  Returns an approximation of the number of threads that are currently running tasks for this executor.
- virtual long long **getCompletedTaskCount** () **const**

*Returns the approximate number of Tasks that have been completed by this **Executor** (p. 1004), this value never decreases.*

- virtual int **getLargestPoolSize** () **const**

  *Returns the most Threads that have ever been active at one time within this **Executors** (p. 1005) Thread pool.*

- virtual **BlockingQueue**
  < **decaf::lang::Runnable** * > * **getQueue** ()

  *Provides access to the Task **Queue** (p. 1723) used by this **Executor** (p. 1004).*

- virtual bool **isTerminating** () **const**

  Returns true if the executor has begin the process of terminating but has not yet completed the process of shutting down all worker threads.

- virtual void **allowCoreThreadTimeout** (bool value)

  When true this setting allows the threads in the core pool to terminate if they sit idle longer than the set keep alive time.

- virtual bool **allowsCoreThreadTimeout** () **const**

  Returns whether this executor has been configured to allow core threads to terminate if they sit idle longer than the configured keep alive time.

- virtual long long **getKeepAliveTime** (**const TimeUnit** &unit) **const**

  Returns the currently set value for the maximum amount of time a worker Thread that is not part of the core threads is allowed to sit idle before it terminates.

- virtual void **setKeepAliveTime** (long long timeout, **const TimeUnit** &unit)

  Configures the amount of time a non core Thread will remain alive after it has completed its assigned task.

- virtual void **setThreadFactory** (**ThreadFactory** *factory)

  *Sets the **ThreadFactory** (p. 2102) instance used to create new Threads for this **Executor** (p. 1004).*

- virtual **ThreadFactory** * **getThreadFactory** () **const**

  *Gets the currently configured **ThreadFactory** (p. 2102).*

- virtual **RejectedExecutionHandler** * **getRejectedExecutionHandler** () **const**

  *Gets the currently configured **RejectedExecutionHandler** (p. 1757) for this **Executor** (p. 1004).*

- virtual void **setRejectedExecutionHandler** (**RejectedExecutionHandler** *handler)

  *Sets the new **RejectedExecutionHandler** (p. 1757) that this executor should use to process any rejected Runnable tasks.*

- virtual bool **prestartCoreThread** ()

  By default a Core thread is only created once the first task is queued, this method forces the creation of core thread that waits in an idle mode for new work to be enqueued.

- virtual int **prestartAllCoreThreads** ()

  This method will create and start new core threads running in an idle state waiting for new tasks up to the set core thread limit.

- bool **remove** (**decaf::lang::Runnable** *task)

  Attempts to remove the Runnable from the work queue, if successful then the caller now owns the Runnable and is responsible for deleting it.

- virtual void **purge** ()

  *Attempts to remove any **Future** (p. 1075) derived tasks from the pending work queue if they have been canceled.*

Protected Member Functions

- virtual void **beforeExecute** (**decaf::lang::Thread** *thread, **decaf::lang::Runnable** *task)

  Method called before a task is executed by the given thread.

- virtual void **afterExecute** (**decaf::lang::Runnable** *task, **decaf::lang::Throwable** *error)

  Called upon completion of execution of a given task.

- virtual void **terminated** ()

  *Method invoked when the **Executor** (p. 1004) has terminated, by default this method does nothing.*

Friends

- class **ExecutorKernel**

6.547.1 Detailed Description

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

The Thread Poll 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 executing it in its thread context.

6.547.2 Constructor & Destructor Documentation

6.547.2.1 decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int *corePoolSize,* int *maxPoolSize,* long long *keepAliveTime,* const TimeUnit & *unit,* BlockingQueue< decaf::lang::Runnable * > * *workQueue*)

Creates a new instance of a **ThreadPoolExecutor** (p. 2104).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

corePoolSize	The number of threads to pool regardless of their idle state.
maxPoolSize	The maximum number of threads that will ever exist at one time in the pool.
keepAliveTime	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
unit	The units that the keepAliveTime is specified in.
workQueue	A **BlockingQueue** (p. 417) implementation that will be used to hold Runnable tasks that are awaiting execution within this executor. The **Executor** (p. 1004) takes ownership of the **BlockingQueue** (p. 417) instance passed once this method returns.

Exceptions

IllegalArguementException	if the corePoolSize or keepAliveTime are negative or the or if maximumPoolSize is less than or equal to zero, or if corePoolSize is greater than maximumPoolSize.
NullPointerException	if the workQueue pointer is NULL.

6.547.2.2 decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int *corePoolSize,* int *maxPoolSize,* long long *keepAliveTime,* const TimeUnit & *unit,* BlockingQueue< decaf::lang::Runnable * > * *workQueue,* RejectedExecutionHandler * *handler*)

Creates a new instance of a **ThreadPoolExecutor** (p. 2104).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

corePoolSize	The number of threads to pool regardless of their idle state.
maxPoolSize	The maximum number of threads that will ever exist at one time in the pool.

keepAliveTime	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
unit	The units that the keepAliveTime is specified in.
workQueue	A **BlockingQueue** (p. 417) implementation that will be used to hold Runnable tasks that are awaiting execution within this executor. The **Executor** (p. 1004) takes ownership of the **BlockingQueue** (p. 417) instance passed once this method returns.
handler	A **RejectedExecutionHandler** (p. 1757) implementation that will be used to handle any rejected tasks when they are submitted to this executor. The **Executor** (p. 1004) takes ownership of the **RejectedExecutionHandler** (p. 1757) instance passed once this method returns.

Exceptions

IllegalArguementException	if the corePoolSize or keepAliveTime are negative or the or if maximumPoolSize is less than or equal to zero, or if corePoolSize is greater than maximumPoolSize.
NullPointerException	if the workQueue pointer is NULL.

6.547.2.3 decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int *corePoolSize,* int *maxPoolSize,* long long *keepAliveTime,* const TimeUnit & *unit,* BlockingQueue< decaf::lang::Runnable * > * *workQueue,* ThreadFactory * *threadFactory*)

Creates a new instance of a **ThreadPoolExecutor** (p. 2104).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

corePoolSize	The number of threads to pool regardless of their idle state.
maxPoolSize	The maximum number of threads that will ever exist at one time in the pool.
keepAliveTime	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
unit	The units that the keepAliveTime is specified in.
workQueue	A **BlockingQueue** (p. 417) implementation that will be used to hold Runnable tasks that are awaiting execution within this executor. The **Executor** (p. 1004) takes ownership of the **BlockingQueue** (p. 417) instance passed once this method returns.
threadFactory	A **ThreadFactory** (p. 2102) implementation that will be used to create worker threads that are used by this executor to run the submitted tasks. The **Executor** (p. 1004) takes ownership of the **ThreadFactory** (p. 2102) instance passed once this method returns.

Exceptions

IllegalArguementException	if the corePoolSize or keepAliveTime are negative or the or if maximumPoolSize is less than or equal to zero, or if corePoolSize is greater than maximumPoolSize.
NullPointerException	if the workQueue pointer is NULL.

6.547.2.4 decaf::util::concurrent::ThreadPoolExecutor::ThreadPoolExecutor (int *corePoolSize,* int *maxPoolSize,* long long *keepAliveTime,* const TimeUnit & *unit,* BlockingQueue< decaf::lang::Runnable * > * *workQueue,* ThreadFactory * *threadFactory,* RejectedExecutionHandler * *handler*)

Creates a new instance of a **ThreadPoolExecutor** (p. 2104).

The executor instance is configured with the passed in parameters and a default thread Factory is used along with a default rejected execution handler.

Parameters

corePoolSize	The number of threads to pool regardless of their idle state.
maxPoolSize	The maximum number of threads that will ever exist at one time in the pool.
keepAliveTime	The maximum time to keep a thread in the pool for if the number of current threads exceeds to core pool size.
unit	The units that the keepAliveTime is specified in.
workQueue	A **BlockingQueue** (p. 417) implementation that will be used to hold Runnable tasks that are awaiting execution within this executor. The **Executor** (p. 1004) takes ownership of the **BlockingQueue** (p. 417) instance passed once this method returns.
threadFactory	A **ThreadFactory** (p. 2102) implementation that will be used to create worker threads that are used by this executor to run the submitted tasks. The **Executor** (p. 1004) takes ownership of the **ThreadFactory** (p. 2102) instance passed once this method returns.
handler	A **RejectedExecutionHandler** (p. 1757) implementation that will be used to handle any rejected tasks when they are submitted to this executor. The **Executor** (p. 1004) takes ownership of the **BlockingQueue** (p. 417) instance passed once this method returns.

Exceptions

IllegalArguementException	if the corePoolSize or keepAliveTime are negative or the or if maximumPoolSize is less than or equal to zero, or if corePoolSize is greater than maximumPoolSize.
NullPointerException	if the workQueue pointer is NULL.

6.547.2.5 virtual decaf::util::concurrent::ThreadPoolExecutor::~ThreadPoolExecutor ( ) [virtual]

6.547.3 Member Function Documentation

6.547.3.1 virtual void decaf::util::concurrent::ThreadPoolExecutor::afterExecute (decaf::lang::Runnable * *task*, decaf::lang::Throwable * *error*) [protected, virtual]

Called upon completion of execution of a given task.

This method is called from the Thread that executed the given Runnable. If the Throwable pointer is not NULL then its value is the Exception that caused the task to terminate.

The base class implementation does nothing, a derived class should call this method on its base class to ensure that all subclasses have a chance to process the afterExecute event.

Parameters

task	The Runnable instance that was executed by the calling Thread.
error	The exception that was thrown from the given Runnable.

6.547.3.2 virtual void decaf::util::concurrent::ThreadPoolExecutor::allowCoreThreadTimeout (bool *value*) [virtual]

When true this setting allows the threads in the core pool to terminate if they sit idle longer than the set keep alive time.

Core threads that terminate are replaced as needed by new ones on demand. This settings requires that the set keep alive time be greater than zero and will throw an IllegalArguementException if that is not the case.

Parameters

value	Boolean value indicating if core threads are allowed to time out when idle.

Exceptions

IllegalArgumentException	if the keep alive time is set to zero.

6.547.3.3 virtual bool decaf::util::concurrent::ThreadPoolExecutor::allowsCoreThreadTimeout ( ) const [virtual]

Returns whether this executor has been configured to allow core threads to terminate if they sit idle longer than the configured keep alive time.

Threads that are not core threads continue to time out using the set keep alive value regardless of whether this option is enabled.

Returns

true if core threads can timeout when idle.

6.547.3.4 virtual bool decaf::util::concurrent::ThreadPoolExecutor::awaitTermination (long long *timeout,* const decaf::util::concurrent::TimeUnit & *unit*) [virtual]

The caller will block until the executor has completed termination meaning all tasks that where scheduled before shutdown have now completed and the executor is ready for deletion.

If the timeout period elapses before the executor reaches the terminated state then this method return false to indicate it has not terminated.

Parameters

timeout	The amount of time to wait before abandoning the wait for termination.
unit	The unit of time that the timeout value represents.

Returns

true if the executor terminated or false if the timeout expired.

Exceptions

InterruptedException	if this call is interrupted while awaiting termination.

Implements **decaf::util::concurrent::ExecutorService** (p. 1009).

6.547.3.5 virtual void decaf::util::concurrent::ThreadPoolExecutor::beforeExecute (decaf::lang::Thread * *thread,* decaf::lang::Runnable * *task*) [protected, virtual]

Method called before a task is executed by the given thread.

The default implementation of this method does nothing, however a subclass can override this method to add some new functionality.

It is recommended that a subclass call this method on its base class to ensure that all base classes have a chance to process this event.

Parameters

thread	The thread that will be executing the given task.
task	The task that will be executed by the given thread.

6.547.3.6 virtual void decaf::util::concurrent::ThreadPoolExecutor::execute (decaf::lang::Runnable * *command*) [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. 1004) implementation.

Parameters

command	the runnable task

Exceptions

RejectedExecution-Exception *(p. 1755)*	if this task cannot be accepted for execution.
NullPointerException	if command is null

Implements **decaf::util::concurrent::Executor** (p. 1005).

Referenced by decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::rejected-Execution().

6.547.3.7 virtual int decaf::util::concurrent::ThreadPoolExecutor::getActiveCount ( ) const [virtual]

Returns an approximation of the number of threads that are currently running tasks for this executor.

This value can change rapidly.

Returns

the number of currently active threads.

6.547.3.8 virtual long long decaf::util::concurrent::ThreadPoolExecutor::getCompletedTaskCount ( ) const [virtual]

Returns the approximate number of Tasks that have been completed by this **Executor** (p. 1004), this value never decreases.

Returns

the number of completed tasks since creation of the **Executor** (p. 1004).

6.547.3.9 virtual int decaf::util::concurrent::ThreadPoolExecutor::getCorePoolSize ( ) const [virtual]

Returns the configured number of core threads for this **Executor** (p. 1004).

Returns

the configured number of core Threads.

6.547.3.10 virtual long long decaf::util::concurrent::ThreadPoolExecutor::getKeepAliveTime (const TimeUnit & *unit*) const [virtual]

Returns the currently set value for the maximum amount of time a worker Thread that is not part of the core threads is allowed to sit idle before it terminates.

Parameters

unit	The unit of time to return the results in.

Returns

the configure keep alive time in the requested time units.

6.547.3.11 virtual int decaf::util::concurrent::ThreadPoolExecutor::getLargestPoolSize ( ) const [virtual]

Returns the most Threads that have ever been active at one time within this **Executors** (p. 1005) Thread pool.

Returns

the largest number of threads ever to coexist in this executor.

6.547.3.12 virtual int decaf::util::concurrent::ThreadPoolExecutor::getMaximumPoolSize ( ) const [virtual]

Returns the configured maximum number of threads for this **Executor** (p. 1004).

Returns

the configured maximum number of Threads.

6.547.3.13 virtual int decaf::util::concurrent::ThreadPoolExecutor::getPoolSize ( ) const [virtual]

Returns the number of threads that currently exists for this **Executor** (p. 1004).

Returns

the configured number of Threads in the Pool.

6.547.3.14 virtual BlockingQueue<decaf::lang::Runnable*>* decaf::util::concurrent::ThreadPoolExecutor::getQueue ( ) [virtual]

Provides access to the Task **Queue** (p. 1723) used by this **Executor** (p. 1004).

This method is meant mainly for debugging and monitoring, care should be taken when using this method. The executor continues to execute tasks from the **Queue** (p. 1723).

Returns

a pointer to the blocking queue that this executor stores future tasks in.

Referenced by decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::rejectedExecution().

6.547.3.15 virtual RejectedExecutionHandler* decaf::util::concurrent::ThreadPoolExecutor::getRejectedExecutionHandler ( ) const [virtual]

Gets the currently configured **RejectedExecutionHandler** (p. 1757) for this **Executor** (p. 1004).

Returns

a pointer to the current **RejectedExecutionHandler** (p. 1757).

6.547.3.16 virtual long long decaf::util::concurrent::ThreadPoolExecutor::getTaskCount ( ) const [virtual]

Returns the current number of pending tasks in the work queue.

This is an approximation as the number of pending tasks can quickly changes as tasks complete and new tasks are started.

Returns

number of outstanding tasks, approximate.

6.547.3.17 virtual ThreadFactory* decaf::util::concurrent::ThreadPoolExecutor::getThreadFactory ( ) const [virtual]

Gets the currently configured **ThreadFactory** (p. 2102).

It is considered a programming error to delete the pointer returned by this method.

Returns

the currently configured **ThreadFactory** (p. 2102) instance used by this object.

6.547.3.18 virtual bool decaf::util::concurrent::ThreadPoolExecutor::isShutdown ( ) const [virtual]

Returns whether this executor has been shutdown or not.

Returns

true if this executor has been shutdown.

Implements **decaf::util::concurrent::ExecutorService** (p. 1009).

Referenced by decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy::rejectedExecution().

6.547.3.19 virtual bool decaf::util::concurrent::ThreadPoolExecutor::isTerminated ( ) const [virtual]

Returns whether all tasks have completed after this executor was shut down.

Returns

true if all tasks have completed after a request to shut down was made.

Implements **decaf::util::concurrent::ExecutorService** (p. 1009).

6.547.3.20 virtual bool decaf::util::concurrent::ThreadPoolExecutor::isTerminating ( ) const [virtual]

Returns true if the executor has begin the process of terminating but has not yet completed the process of shutting down all worker threads.

If the **Executor** (p. 1004) does not transition from this state to terminated after some time its generally an indication that one of the submitted tasks will not complete and the executor is locked in a terminating state.

Returns

true if all tasks have completed after a request to shut down was made.

6.547.3.21 virtual int decaf::util::concurrent::ThreadPoolExecutor::prestartAllCoreThreads ( ) [virtual]

This method will create and start new core threads running in an idle state waiting for new tasks up to the set core thread limit.

When the limit is reached this method returns zero to indicate no more core threads can be created.

Returns

the number of core threads created, or zero if the limit has already been met.

6.547.3.22 virtual bool decaf::util::concurrent::ThreadPoolExecutor::prestartCoreThread ( ) [virtual]

By default a Core thread is only created once the first task is queued, this method forces the creation of core thread that waits in an idle mode for new work to be enqueued.

If the limit on core threads has already been reached then this method returns false.

Returns

true if a new core thread was added, false otherwise.

6.547.3.23 virtual void decaf::util::concurrent::ThreadPoolExecutor::purge ( ) [virtual]

Attempts to remove any **Future** (p. 1075) derived tasks from the pending work queue if they have been canceled.

This method can be used to more quickly remove and reclaim space as canceled tasks are not run but must await a worker thread to be removed normally. Since there are multiple threads in operation its possible for this method to not remove all canceled tasks from the work queue.

6.547.3.24 bool decaf::util::concurrent::ThreadPoolExecutor::remove (decaf::lang::Runnable * *task*)

Attempts to remove the Runnable from the work queue, if successful then the caller now owns the Runnable and is responsible for deleting it.

Parameters

task	The task that is to be removed from the work queue.

Returns

true if the task was removed from the **Queue** (p. 1723).

6.547.3.25 virtual void decaf::util::concurrent::ThreadPoolExecutor::setCorePoolSize (int *poolSize*) [virtual]

Set (p. 1857) the number of threads that this executor treats as its core threads, this value will override the value set in the constructor.

If the value given is less than the current value then the core threads will shrink to the new value over time. If the value is larger than the current value then new threads may be started to process currently pending tasks, otherwise they will be started as needed when new tasks arrive.

Parameters

poolSize	The new core pool size for this executor.

Exceptions

IllegalArgumentException	if the pool size value is less than zero.

6.547.3.26 virtual void decaf::util::concurrent::ThreadPoolExecutor::setKeepAliveTime (long long *timeout*, const TimeUnit & *unit*) [virtual]

Configures the amount of time a non core Thread will remain alive after it has completed its assigned task.

This value can also be applied to core threads if the allowCoreThreadsTimeout option is enabled.

Parameters

timeout	The amount of time an idle worker will live before terminating.
unit	The units that the timeout is given in.

Exceptions

IllegalArgumentException	if allowCoreThreadsTimeout is enabled and the the timeout value given is zero, or the timeout given is negative.

6.547.3.27 virtual void decaf::util::concurrent::ThreadPoolExecutor::setMaximumPoolSize (int *maxSize*) [virtual]

Sets the maximum number of workers this **Executor** (p. 1004) is allowed to have at any given time above the core pool size.

This new value overrides any set in the constructor and if smaller than the current value worker threads will terminate as they complete their current tasks and become idle.

Parameters

maxSize	The new maximum allowed worker pool size.

Exceptions

IllegalArgumentException	if maxSize is negative or less than core pool size.

6.547.3.28 virtual void decaf::util::concurrent::ThreadPoolExecutor::setRejectedExecutionHandler (RejectedExecutionHandler * *handler*) [virtual]

Sets the new **RejectedExecutionHandler** (p. 1757) that this executor should use to process any rejected Runnable tasks.

This executor takes ownership of the supplied pointer and will desotroy it upon termination, any previous handler is destroyed by this call.

Parameters

handler	The new **RejectedExecutionHandler** (p. 1757) instance to use.

Exceptions

NullPointerException	if the handler is NULL.

6.547.3.29 virtual void decaf::util::concurrent::ThreadPoolExecutor::setThreadFactory (ThreadFactory * *factory*) [virtual]

Sets the **ThreadFactory** (p. 2102) instance used to create new Threads for this **Executor** (p. 1004).

This class takes ownership of the given **ThreadFactory** (p. 2102) and will destroy it upon termination or when a new **ThreadFactory** (p. 2102) is set using this method.

Parameters

factory	A **ThreadFactory** (p. 2102) instance used by this **Executor** (p. 1004) to create new Threads.

Exceptions

NullPointerException	if the given factory pointer is NULL.

6.547.3.30 virtual void decaf::util::concurrent::ThreadPoolExecutor::shutdown ( ) [virtual]

Performs an orderly shutdown of this **Executor** (p. 1004).

Previously queued tasks are allowed to complete but no new tasks are accepted for execution. Calling this method more than once has no affect on this executor.

Implements **decaf::util::concurrent::ExecutorService** (p. 1009).

6.547.3.31 virtual ArrayList<decaf::lang::Runnable*> decaf::util::concurrent::ThreadPoolExecutor::shutdownNow ( ) [virtual]

Attempts to stop all currently executing tasks and returns an **ArrayList** (p. 345) containing the Runnables that did not get executed, these object become the property of the caller and are not deleted by this class, they are removed from the work queue and forgotten about.

There is no guarantee that this method will halt execution of currently executing tasks.

Returns

an **ArrayList** (p. 345) containing all Runnable instance that were still waiting to be executed by this class, call now owns those pointers.

Implements **decaf::util::concurrent::ExecutorService** (p. 1010).

6.547.3.32 virtual void decaf::util::concurrent::ThreadPoolExecutor::terminated ( ) [protected, virtual]

Method invoked when the **Executor** (p. 1004) has terminated, by default this method does nothing.

When overridden the subclass should call superclass::terminated to ensure that all subclasses have their terminated method invoked.

6.547.4 Friends And Related Function Documentation

6.547.4.1 friend class ExecutorKernel [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPoolExecutor.h**

6.548 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.

6.548.1 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. 990) class in order to ensure that all Decaf Exceptions are interchangeable with the std::exception class.

Throwable (p. 2116) can wrap another **Throwable** (p. 2116) 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. 2116) instance and will be deleted when it is deleted or goes out of scope.

Since

1.0

6.548.2 Constructor & Destructor Documentation

6.548.2.1 decaf::lang::Throwable::Throwable ( ) throw () [inline]

6.548.2.2 virtual decaf::lang::Throwable::~Throwable ( ) throw () [inline, virtual]

6.548.3 Member Function Documentation

6.548.3.1 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. 990) object

Implemented in **decaf::net::URISyntaxException** (p. 2216), **decaf::lang::Exception** (p. 992), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 1574), **decaf::lang::exceptions::InvalidStateException** (p. 1198), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1098), **decaf::lang::exceptions::InterruptedException** (p. 1187), **decaf::security::GeneralSecurityException** (p. 1081), **decaf::security::NoSuchProviderException** (p. 1541), **decaf::security::NoSuchAlgorithmException** (p. 1537), **decaf::security::SignatureException** (p. 1891), **decaf::lang::exceptions::IllegalStateException** (p. 1101), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1103), **decaf::lang::exceptions::NullPointerException** (p. 1543), **decaf::security::InvalidKeyException** (p. 1193), **decaf::lang::exceptions::IllegalArgumentException** (p. 1096), **decaf::util::concurrent::BrokenBarrierException** (p. 432), **decaf::lang::exceptions::RuntimeException** (p. 1796), **decaf::security::KeyException** (p. 1242), **decaf::security::KeyManagementException** (p. 1245), **decaf::util::concurrent::ExecutionException** (p. 1003), **decaf::util::concurrent::TimeoutException** (p. 2122), **decaf::util::NoSuchElementException** (p. 1539), **decaf::lang::exceptions::ClassCastException** (p. 632), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1108), **decaf::lang::exceptions::NumberFormatException** (p. 1547), **decaf::util::concurrent::CancellationException** (p. 582), **decaf::util::concurrent::RejectedExecutionException** (p. 1757), **decaf::net::BindException** (p. 413), **decaf::lang::exceptions::UnsupportedOperationException** (p. 2189), **decaf::net::ConnectException** (p. 724), **decaf::nio::InvalidMarkException** (p. 1195), **decaf::io::UTFDataFormatException** (p. 2230), **decaf::io::UnsupportedEncodingException** (p. 2187), **decaf::net::HttpRetryException** (p. 1092), **decaf::net::MalformedURLException** (p. 1371), **decaf::net::NoRouteToHostException** (p. 1534), **decaf::net::PortUnreachableException** (p. 1634), **decaf::net::ProtocolException** (p. 1715), **decaf::net::SocketTimeoutException** (p. 1934), **decaf::net::UnknownHostException** (p. 2183), **decaf::net::UnknownServiceException** (p. 2185), **decaf::util::zip::ZipException** (p. 2292), **decaf::io::EOFException** (p. 988), **decaf::io::InterruptedIOException** (p. 1189), **decaf::nio::ReadOnlyBufferException** (p. 1741), **decaf::util::ConcurrentModificationException** (p. 701), **decaf::util::zip::DataFormatException** (p. 835), **decaf::io::IOException** (p. 1200), **decaf::nio::BufferOverflowException** (p. 474), **decaf::nio::BufferUnderflowException** (p. 476), **decaf::net::SocketException** (p. 1916), **decaf::security::cert::CertificateExpiredException** (p. 590), **decaf::security::cert::CertificateNotYetValidException** (p. 591), **decaf::security::cert::CertificateParsingException** (p. 593), **decaf::security::cert::CertificateEncodingException** (p. 587), **decaf::security::cert::CertificateException** (p. 588), **activemq::exceptions::ActiveMQException** (p. 204), **activemq::exceptions::BrokerException** (p. 437), and **activemq::exceptions::ConnectionFailedException** (p. 745).

6.548.3.2 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. 993).

6.548.3.3 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. 993).

6.548.3.4 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. 994).

6.548.3.5 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. 994).

6.548.3.6 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. 994).

6.548.3.7 virtual void decaf::lang::Throwable::printStackTrace ( ) const [pure virtual]

Prints the stack trace to std::err.

Implemented in **decaf::lang::Exception** (p. 994).

6.548.3.8 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. 994).

6.548.3.9 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. 995).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Throwable.h**

6.549 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 ()

6.549.1 Constructor & Destructor Documentation

6.549.1.1 decaf::util::concurrent::TimeoutException::TimeoutException ( ) throw () [inline]

Default Constructor.

6.549.1.2 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

6.549.1.3 decaf::util::concurrent::TimeoutException::TimeoutException (const TimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	The exception to copy from.

6.549.1.4 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.

6.549.1.5 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

6.549.1.6 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

6.549.1.7 virtual decaf::util::concurrent::TimeoutException::~TimeoutException ( ) throw () [inline, virtual]

6.549.2 Member Function Documentation

6.549.2.1 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. 2120) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TimeoutException.h**

6.550 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** ()
- **Timer** (**const** std::string &name)

  *Create a new **Timer** (p. 2122) whose associated thread is assigned the name given.*
- virtual ~**Timer** ()
- void **cancel** ()

  Terminates this timer, discarding any currently scheduled tasks.
- int **purge** ()

  Removes all canceled tasks from this timer's task queue.
- void **schedule** (**TimerTask** *task, long long delay)

  Schedules the specified task for execution after the specified delay.
- void **schedule** (**const decaf::lang::Pointer**< **TimerTask** > &task, long long delay)

  Schedules the specified task for execution after the specified delay.
- void **schedule** (**TimerTask** *task, **const Date** &time)

  Schedules the specified task for execution at the specified time.
- void **schedule** (**const decaf::lang::Pointer**< **TimerTask** > &task, **const Date** &time)

Schedules the specified task for execution at the specified time.

- void **schedule** (**TimerTask** ∗task, long long delay, long long period)

  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)

  Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.
- void **schedule** (**TimerTask** ∗task, **const Date** &firstTime, long long period)

  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)

  Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.
- void **scheduleAtFixedRate** (**TimerTask** ∗task, long long delay, long long period)

  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)

  Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.
- void **scheduleAtFixedRate** (**TimerTask** ∗task, **const Date** &firstTime, long long period)

  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)

  Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

6.550.1 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. 2122) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 2122) 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. 2122) 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

6.550.2 Constructor & Destructor Documentation

6.550.2.1 decaf::util::Timer::Timer ( )

6.550.2.2 decaf::util::Timer::Timer (const std::string & *name*)

Create a new **Timer** (p. 2122) whose associated thread is assigned the name given.

Parameters

name	The name to assign to this **Timer** (p. 2122)'s Thread.

6.550.2.3 virtual decaf::util::Timer::∼Timer ( ) [virtual]

6.550.3 Member Function Documentation

6.550.3.1 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.

6.550.3.2 int 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. 2122) to destroy the **TimerTask** (p. 2130) 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. 2122) object that has no scheduled tasks without error.

Returns

the number of tasks removed from the queue.

6.550.3.3 void decaf::util::Timer::schedule (TimerTask * *task,* long long *delay*)

Schedules the specified task for execution after the specified delay.

The **TimerTask** (p. 2130) pointer is considered to be owned by the **Timer** (p. 2122) class once it has been scheduled, the **Timer** (p. 2122) will destroy its **TimerTask** (p. 2130)'s once they have been canceled or the **Timer** (p. 2122) itself is canceled. A **TimerTask** (p. 2130) 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. 2122) and the caller should ensure that the **TimerTask** (p. 2130) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2130) 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. 2130) value is Null.
IllegalArgumentException	- if delay is negative, or delay + **System.currentTimeMillis()** (p. 2070) is negative.
IllegalStateException	- if task was already scheduled or canceled, or timer was canceled.

6.550.3.4 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task,* long long *delay*)

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. 2130) value is Null.
IllegalArgumentException	- if delay is negative, or delay + **System.currentTimeMillis()** (p. 2070) is negative.
IllegalStateException	- if task was already scheduled or canceled, or timer was canceled.

6.550.3.5 void decaf::util::Timer::schedule (TimerTask * *task,* const Date & *time*)

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. 2130) pointer is considered to be owned by the **Timer** (p. 2122) class once it has been scheduled, the **Timer** (p. 2122) will destroy its **TimerTask** (p. 2130)'s once they have been canceled or the **Timer** (p. 2122) itself is canceled. A **TimerTask** (p. 2130) 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. 2122) and the caller should ensure that the **TimerTask** (p. 2130) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2130) instance are planned.

Parameters

task	- task to be scheduled.
time	- time at which task is to be executed.

Exceptions

NullPointerException	- if the **TimerTask** (p. 2130) value is Null.
IllegalArgumentException	- if time.getTime() is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.6 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task,* const Date & *time*)

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. 2130) value is Null.
IllegalArgumentException	- if time.getTime() is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.7 void decaf::util::Timer::schedule (TimerTask * *task,* long long *delay,* long long *period*)

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. 2130) pointer is considered to be owned by the **Timer** (p. 2122) class once it has been scheduled, the **Timer** (p. 2122) will destroy its **TimerTask** (p. 2130)'s once they have been canceled or the **Timer** (p. 2122) itself is canceled. A **TimerTask** (p. 2130) 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. 2122) and the caller should ensure that the **TimerTask** (p. 2130) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2130) 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. 2130) value is Null.
IllegalArgumentException	- if delay is negative, or delay + **System.currentTimeMillis()** (p. 2070) is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.8 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task,* long long *delay,* long long *period*)

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. 2130) value is Null.
IllegalArgumentException	- if delay is negative, or delay + **System.currentTimeMillis()** (p. 2070) is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.9 void decaf::util::Timer::schedule (TimerTask * *task,* const Date & *firstTime,* long long *period*)

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. 2130) pointer is considered to be owned by the **Timer** (p. 2122) class once it has been scheduled, the **Timer** (p. 2122) will destroy its **TimerTask** (p. 2130)'s once they have been canceled or the **Timer** (p. 2122) itself is canceled. A **TimerTask** (p. 2130) 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. 2122) and the caller should ensure that the **TimerTask** (p. 2130) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2130) 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. 2130) value is Null.
IllegalArgumentException	- if time.getTime() is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.10 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task,* const Date & *firstTime,* long long *period*)

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. 2130) value is Null.
IllegalArgumentException	- if time.getTime() is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.11 void decaf::util::Timer::scheduleAtFixedRate (TimerTask * *task,* long long *delay,* long long *period*)

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. 2130) pointer is considered to be owned by the **Timer** (p. 2122) class once it has been scheduled, the **Timer** (p. 2122) will destroy its **TimerTask** (p. 2130)'s once they have been canceled or the **Timer** (p. 2122) itself is canceled. A **TimerTask** (p. 2130) 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. 2122) and the caller should ensure that the **TimerTask** (p. 2130) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2130) 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 count down 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. 2130) value is Null.
IllegalArgumentException	- if delay is negative, or delay + **System.currentTimeMillis()** (p. 2070) is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.12 void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & *task,* long long *delay,* long long *period*)

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. 2130) value is Null.
IllegalArgumentException	- if delay is negative, or delay + **System.currentTimeMillis()** (p. 2070) is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.13 void decaf::util::Timer::scheduleAtFixedRate (TimerTask * *task,* const Date & *firstTime,* long long *period*)

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. 2130) pointer is considered to be owned by the **Timer** (p. 2122) class once it has been scheduled, the **Timer** (p. 2122) will destroy its **TimerTask** (p. 2130)'s once they have been canceled or the **Timer** (p. 2122) itself is canceled. A **TimerTask** (p. 2130) 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. 2122) and the caller should ensure that the **TimerTask** (p. 2130) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 2130) 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. 2130) value is Null.

IllegalArgumentException	- if time.getTime() is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

6.550.3.14 void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & *task,* const Date & *firstTime,* long long *period*)

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. 2130) value is Null.
IllegalArgumentException	- if time.getTime() is negative.
IllegalStateException	- if task was already scheduled or canceled, timer was canceled, or timer thread terminated.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Timer.h**

6.551 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. 2122).

`#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**

6.551.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2122).

Since

1.0

6.551.2 Constructor & Destructor Documentation

6.551.2.1 decaf::util::TimerTask::TimerTask ( )

6.551.2.2 virtual decaf::util::TimerTask::~TimerTask ( ) [inline, virtual]

6.551.3 Member Function Documentation

6.551.3.1 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.)

6.551.3.2 long long decaf::util::TimerTask::getWhen ( ) const [protected]

6.551.3.3 bool decaf::util::TimerTask::isScheduled ( ) const [protected]

6.551.3.4 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() {
    if( System::currentTimeMillis() - scheduledExecutionTime() >=
        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. 884). The return value is undefined if the task has yet to commence its first execution.

6.551.3.5 void decaf::util::TimerTask::setScheduledTime (long long *time*) [protected]

6.551.4 Friends And Related Function Documentation

6.551.4.1 friend class decaf::internal::util::TimerTaskHeap [friend]

6.551.4.2 friend class Timer [friend]

6.551.4.3 friend class TimerImpl [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**TimerTask.h**

6.552 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** ()

  Peaks 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 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.

6.552.1 Detailed Description

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

Since

1.0

6.552.2 Constructor & Destructor Documentation

6.552.2.1 decaf::internal::util::TimerTaskHeap::TimerTaskHeap ( )

6.552.2.2 virtual decaf::internal::util::TimerTaskHeap::~TimerTaskHeap ( ) [virtual]

6.552.3 Member Function Documentation

6.552.3.1 void decaf::internal::util::TimerTaskHeap::adjustMinimum ( )

Resorts the heap starting at the top.

6.552.3.2 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 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 becuase they were cancelled.

6.552.3.3 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.

6.552.3.4 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.

6.552.3.5 bool decaf::internal::util::TimerTaskHeap::isEmpty ( ) const

Returns

true if the heap is empty.

6.552.3.6 Pointer<TimerTask> decaf::internal::util::TimerTaskHeap::peek ( )

Peaks 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.

6.552.3.7 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.

6.552.3.8 void decaf::internal::util::TimerTaskHeap::reset ( )

Clear all contents from the heap.

6.552.3.9 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**

6.553 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 2134) 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**

  Perform a timed `Object.wait` *using this time unit.*
- void **timedJoin** (**decaf::lang::Thread** *thread, long long timeout)

  Perform a timed `Thread.join` *using this time unit.*
- void **sleep** (long long timeout) **const**

  Perform a `Thread.sleep` *using this unit.*
- virtual std::string **toString** () **const**

  *Converts the **TimeUnit** (p. 2134) type to the Name of the **TimeUnit** (p. 2134).*
- virtual int **compareTo** (**const TimeUnit** &value) **const**
- virtual bool **equals** (**const TimeUnit** &value) **const**
- virtual bool **operator==** (**const TimeUnit** &value) **const**
- virtual bool **operator<** (**const TimeUnit** &value) **const**

Static Public Member Functions

- static **const TimeUnit** & **valueOf** (**const** std::string &name)

  *Returns the **TimeUnit** (p. 2134) constant of this type with the specified name.*

Static Public Attributes

- static **const TimeUnit NANOSECONDS**

  *The Actual **TimeUnit** (p. 2134) 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. 2134) Instances.*

Protected Member Functions

- **TimeUnit** (int index, **const** std::string &name)

  Hidden Constructor, this class can not be instantiated directly.

6.553.1 Detailed Description

A **TimeUnit** (p. 2134) 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. 2134) 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. 2134) 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. 1304) lock = ...; if (lock.tryLock(50, **TimeUnit.MILLISECONDS** (p. 2141))) ...

while this code will timeout in 50 seconds:

Lock (p. 1304) lock = ...; if (lock.tryLock(50, **TimeUnit.SECONDS** (p. 2141))) ...

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. 2134).

6.553.2 Constructor & Destructor Documentation

6.553.2.1 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.

6.553.2.2 virtual decaf::util::concurrent::TimeUnit::~TimeUnit ( ) [inline, virtual]

6.553.3 Member Function Documentation

6.553.3.1 virtual int decaf::util::concurrent::TimeUnit::compareTo (const TimeUnit & *value*) const [virtual]

6.553.3.2 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. 2141))

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.

6.553.3.3 virtual bool decaf::util::concurrent::TimeUnit::equals (const TimeUnit & *value*) const [virtual]

6.553.3.4 virtual bool decaf::util::concurrent::TimeUnit::operator< (const TimeUnit & *value*) const [virtual]

6.553.3.5 virtual bool decaf::util::concurrent::TimeUnit::operator== (const TimeUnit & *value*) const [virtual]

6.553.3.6 void decaf::util::concurrent::TimeUnit::sleep (long long *timeout*) const

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

6.553.3.7 void decaf::util::concurrent::TimeUnit::timedJoin (decaf::lang::Thread * *thread,* long long *timeout*)

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)

6.553.3.8 void decaf::util::concurrent::TimeUnit::timedWait (Synchronizable * *obj,* long long *timeout*) const

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. 421)) 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. 2045) object is null.

See also

Synchronizable::wait(long long, long long)

6.553.3.9 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. 2136)

6.553.3.10 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. 2136)

6.553.3.11 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. 2136)

6.553.3.12 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. 2136)

Referenced by decaf::util::concurrent::LinkedBlockingQueue< E >::offer(), and decaf::util::concurrent::LinkedBlockingQueue< E >::poll().

6.553.3.13 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. 2136)

6.553.3.14 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. 2136)

6.553.3.15 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. 2136)

6.553.3.16 virtual std::string decaf::util::concurrent::TimeUnit::toString ( ) const [virtual]

Converts the **TimeUnit** (p. 2134) type to the Name of the **TimeUnit** (p. 2134).

Returns

String name of the **TimeUnit** (p. 2134)

6.553.3.17 static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf (const std::string & *name*) [static]

Returns the **TimeUnit** (p. 2134) constant of this type with the specified name.

The string must match exactly an identifier used to declare an **TimeUnit** (p. 2134) constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters

name	The Name of the **TimeUnit** (p. 2134) constant to be returned.

Returns

A constant reference to the **TimeUnit** (p. 2134) Constant with the given name.

Exceptions

IllegalArgumentException	if this enum type has no constant with the specified name

6.553.4 Field Documentation

6.553.4.1 const TimeUnit decaf::util::concurrent::TimeUnit::DAYS [static]

6.553.4.2 const TimeUnit decaf::util::concurrent::TimeUnit::HOURS [static]

6.553.4.3 const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS [static]

6.553.4.4 const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS [static]

6.553.4.5 const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES [static]

6.553.4.6 const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS [static]

The Actual **TimeUnit** (p. 2134) enumerations.

6.553.4.7 const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS [static]

6.553.4.8 const TimeUnit* const decaf::util::concurrent::TimeUnit::values[] [static]

The An Array of **TimeUnit** (p. 2134) Instances.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TimeUnit.h**

6.554 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** () throw ()
- virtual std::string **getTopicName** () **const** =0

  Gets the name of this topic.

6.554.1 Detailed Description

An interface encapsulating a provider-specific topic name.

A **Topic** (p. 2141) is a Publish / Subscribe type **Destination** (p. 936). All Messages sent to a **Topic** (p. 2141) are broadcast to all Subscribers of that **Topic** (p. 2141) unless the Subscriber defines a **Message** (p. 1426) selector that filters out that **Message** (p. 1426).

Since

1.0

6.554.2 Constructor & Destructor Documentation

6.554.2.1 virtual cms::Topic::~Topic ( ) throw () [virtual]

6.554.3 Member Function Documentation

6.554.3.1 virtual std::string cms::Topic::getTopicName ( ) const [pure virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSException *(p. 640)*	- If an internal error occurs.

Implemented in **activemq::commands::ActiveMQTopic** (p. 324).

The documentation for this class was generated from the following file:

- src/main/cms/**Topic.h**

6.555 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**

6.555.1 Constructor & Destructor Documentation

6.555.1.1 activemq::state::Tracked::Tracked ( )

6.555.1.2 activemq::state::Tracked::Tracked (const Pointer< decaf::lang::Runnable > & *runnable*)

6.555.1.3 virtual activemq::state::Tracked::~Tracked ( ) [inline, virtual]

6.555.2 Member Function Documentation

6.555.2.1 bool activemq::state::Tracked::isWaitingForResponse ( ) const [inline]

6.555.2.2 void activemq::state::Tracked::onResponse ( )

The documentation for this class was generated from the following file:

- src/main/activemq/state/**Tracked.h**

6.556 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **TransactionId** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual bool **isLocalTransactionId** () **const**
- virtual bool **isXATransactionId** () **const**
- 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

6.556.1 Member Typedef Documentation

6.556.1.1 typedef decaf::lang::PointerComparator<TransactionId> activemq::commands::TransactionId::COMPARATOR

Reimplemented in **activemq::commands::XATransactionId** (p. 2278), and **activemq::commands::LocalTransactionId** (p. 1298).

6.556.2 Constructor & Destructor Documentation

6.556.2.1 activemq::commands::TransactionId::TransactionId ( )

6.556.2.2 activemq::commands::TransactionId::TransactionId (const TransactionId & *other*)

6.556.2.3 virtual activemq::commands::TransactionId::~TransactionId ( ) [virtual]

6.556.3 Member Function Documentation

6.556.3.1 virtual TransactionId* activemq::commands::TransactionId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

Reimplemented in **activemq::commands::XATransactionId** (p. 2278), and **activemq::commands::LocalTransactionId** (p. 1299).

6.556.3.2 virtual int activemq::commands::TransactionId::compareTo (const TransactionId & *value*) const [virtual]

6.556.3.3 virtual void activemq::commands::TransactionId::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. 879).

Reimplemented in **activemq::commands::XATransactionId** (p. 2279), and **activemq::commands::LocalTransactionId** (p. 1299).

6.556.3.4 virtual bool activemq::commands::TransactionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Implements **activemq::commands::DataStructure** (p. 879).

Reimplemented in **activemq::commands::XATransactionId** (p. 2279), and **activemq::commands::LocalTransactionId** (p. 1299).

6.556.3.5 virtual bool activemq::commands::TransactionId::equals (const TransactionId & *value*) const [virtual]

6.556.3.6 virtual unsigned char activemq::commands::TransactionId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

Reimplemented in **activemq::commands::XATransactionId** (p. 2280), and **activemq::commands::LocalTransactionId** (p. 1300).

6.556.3.7 virtual bool activemq::commands::TransactionId::isLocalTransactionId ( ) const [inline, virtual]

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1300).

6.556.3.8 virtual bool activemq::commands::TransactionId::isXATransactionId ( ) const [inline, virtual]

Reimplemented in **activemq::commands::XATransactionId** (p. 2281).

6.556.3.9 virtual bool activemq::commands::TransactionId::operator< (const TransactionId & *value*) const [virtual]

6.556.3.10 TransactionId& activemq::commands::TransactionId::operator= (const TransactionId & *other*)

6.556.3.11 virtual bool activemq::commands::TransactionId::operator== (const TransactionId & *value*) const [virtual]

6.556.3.12 virtual std::string activemq::commands::TransactionId::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 411).

Reimplemented in **activemq::commands::XATransactionId** (p. 2281), and **activemq::commands::LocalTransactionId** (p. 1300).

6.556.4 Field Documentation

6.556.4.1 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**

6.557 activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2146).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/TransactionIdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual ~**TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.557.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (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

6.557.2 Constructor & Destructor Documentation

6.557.2.1 activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]

6.557.2.2 virtual activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::~TransactionIdMarshaller ( ) [inline, virtual]

6.557.3 Member Function Documentation

6.557.3.1 virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1302), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2283).

6.557.3.2 virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1302), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2283).

6.557.3.3 virtual int activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1303), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2283).

6.557.3.4 virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1303), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2284).

6.557.3.5 virtual void activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

Reimplemented in **activemq::wireformat::openwire::marshal::generated::LocalTransactionIdMarshaller** (p. 1303), and **activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller** (p. 2284).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**TransactionIdMarshaller.h**

6.558 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **TransactionInfo** * **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** *value) **const**

  *Compares the **DataStructure** (p. 877) 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)

  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 Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**

6.558.1 Constructor & Destructor Documentation

6.558.1.1 activemq::commands::TransactionInfo::TransactionInfo ( )

6.558.1.2 virtual activemq::commands::TransactionInfo::~TransactionInfo ( ) [virtual]

6.558.2 Member Function Documentation

6.558.2.1 virtual TransactionInfo* activemq::commands::TransactionInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.558.2.2 virtual void activemq::commands::TransactionInfo::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. 382).

6.558.2.3 virtual bool activemq::commands::TransactionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.558.2.4 virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ( ) const [virtual]

6.558.2.5 virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ( ) [virtual]

6.558.2.6 virtual unsigned char activemq::commands::TransactionInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.558.2.7 virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ( ) const [virtual]

6.558.2.8 virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ( ) [virtual]

6.558.2.9 virtual unsigned char activemq::commands::TransactionInfo::getType ( ) const [virtual]

Referenced by activemq::state::CommandVisitorAdapter::processTransactionInfo().

6.558.2.10 virtual bool activemq::commands::TransactionInfo::isTransactionInfo ( ) const [inline, virtual]

Returns

an answer of true to the **isTransactionInfo()** (p. 2151) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 385).

6.558.2.11 virtual void activemq::commands::TransactionInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.558.2.12 virtual void activemq::commands::TransactionInfo::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.558.2.13 virtual void activemq::commands::TransactionInfo::setType (unsigned char *type*) [virtual]

6.558.2.14 virtual std::string activemq::commands::TransactionInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.558.2.15 virtual Pointer<Command> activemq::commands::TransactionInfo::visit (activemq::state::CommandVisitor * *visitor*) [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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.558.3 Field Documentation

6.558.3.1 Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId [protected]

6.558.3.2 const unsigned char activemq::commands::TransactionInfo::ID_TRANSACTIONINFO = 7 [static]

6.558.3.3 Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId [protected]

6.558.3.4 unsigned char activemq::commands::TransactionInfo::type [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionInfo.h**

6.559 activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2152).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/TransactionInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual ~**TransactionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)

  Tight Marhsal to the given stream.

6.559.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2152).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.559.2 Constructor & Destructor Documentation

6.559.2.1 activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::TransactionInfoMarshaller ( ) [inline]

6.559.2.2 virtual activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::~TransactionInfoMarshaller ( ) [inline, virtual]

6.559.3 Member Function Documentation

6.559.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.559.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.559.3.3 virtual void activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 387).

6.559.3.4 virtual void activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 388).

6.559.3.5 virtual int activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 389).

6.559.3.6 virtual void activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 390).

6.559.3.7 virtual void activemq::wireformat::openwire::marshal::generated::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::BaseCommandMarshaller** (p. 391).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**TransactionInfoMarshaller.h**

6.560 cms::TransactionInProgressException Class Reference

This exception is thrown when an operation is invalid because a transaction is in progress.

`#include <src/main/cms/TransactionInProgressException.h>`

Inheritance diagram for cms::TransactionInProgressException:

Public Member Functions

- **TransactionInProgressException** ()
- **TransactionInProgressException** (**const TransactionInProgressException** &ex)
- **TransactionInProgressException** (**const** std::string &message)
- **TransactionInProgressException** (**const** std::string &message, **const** std::exception *cause)
- **TransactionInProgressException** (**const** std::string &message, **const** std::exception *cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**TransactionInProgressException** () throw ()

6.560.1 Detailed Description

This exception is thrown when an operation is invalid because a transaction is in progress.

For instance, an attempt to call **Session::commit** (p. 1834) when a session is part of a distributed transaction should throw a **TransactionInProgressException** (p. 2155).

Since

2.3

6.560.2 Constructor & Destructor Documentation

6.560.2.1 cms::TransactionInProgressException::TransactionInProgressException ( )

6.560.2.2 cms::TransactionInProgressException::TransactionInProgressException (const TransactionInProgressException & *ex*)

6.560.2.3 cms::TransactionInProgressException::TransactionInProgressException (const std::string & *message*)

6.560.2.4 cms::TransactionInProgressException::TransactionInProgressException (const std::string & *message,* const std::exception * *cause*)

6.560.2.5 cms::TransactionInProgressException::TransactionInProgressException (const std::string & *message,* const std::exception * *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.560.2.6 virtual cms::TransactionInProgressException::~TransactionInProgressException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**TransactionInProgressException.h**

6.561 cms::TransactionRolledBackException Class Reference

This exception must be thrown when a call to **Session.commit** (p. 1834) results in a rollback of the current transaction.

`#include <src/main/cms/TransactionRolledBackException.h>`

Inheritance diagram for cms::TransactionRolledBackException:

Public Member Functions

- **TransactionRolledBackException** ()
- **TransactionRolledBackException** (**const TransactionRolledBackException** &ex)
- **TransactionRolledBackException** (**const** std::string &message)
- **TransactionRolledBackException** (**const** std::string &message, **const** std::exception *cause)
- **TransactionRolledBackException** (**const** std::string &message, **const** std::exception *cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**TransactionRolledBackException** () throw ()

6.561.1 Detailed Description

This exception must be thrown when a call to **Session.commit** (p. 1834) results in a rollback of the current transaction.

Since

2.3

6.561.2 Constructor & Destructor Documentation

6.561.2.1 cms::TransactionRolledBackException::TransactionRolledBackException ( )

6.561.2.2 cms::TransactionRolledBackException::TransactionRolledBackException (const TransactionRolledBackException & *ex*)

6.561.2.3 cms::TransactionRolledBackException::TransactionRolledBackException (const std::string & *message*)

6.561.2.4 cms::TransactionRolledBackException::TransactionRolledBackException (const std::string & *message*, const std::exception * *cause*)

6.561.2.5 cms::TransactionRolledBackException::TransactionRolledBackException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.561.2.6 virtual cms::TransactionRolledBackException::~TransactionRolledBackException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**TransactionRolledBackException.h**

6.562 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 LinkedList< Pointer < Command > >** & **getCommands** () **const**
- **const Pointer< TransactionId >** & **getId** () **const**
- void **setPrepared** (bool prepared)
- bool **isPrepared** () **const**
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () **const**
- void **addProducerState** (**const Pointer< ProducerState >** &producerState)
- std::vector< **Pointer < ProducerState > > getProducerStates** ()

6.562.1 Constructor & Destructor Documentation

6.562.1.1 activemq::state::TransactionState::TransactionState (const Pointer< TransactionId > & *id*)

6.562.1.2 virtual activemq::state::TransactionState::~TransactionState ( ) [virtual]

6.562.2 Member Function Documentation

6.562.2.1 void activemq::state::TransactionState::addCommand (const Pointer< Command > & *operation*)

6.562.2.2 void activemq::state::TransactionState::addProducerState (const Pointer< ProducerState > & *producerState*)

6.562.2.3 void activemq::state::TransactionState::checkShutdown ( ) const

6.562.2.4 const LinkedList< Pointer<Command> >& activemq::state::TransactionState::getCommands ( ) const [inline]

6.562.2.5 const Pointer<TransactionId>& activemq::state::TransactionState::getId ( ) const [inline]

6.562.2.6 int activemq::state::TransactionState::getPreparedResult ( ) const [inline]

6.562.2.7 std::vector< Pointer<ProducerState> > activemq::state::TransactionState::getProducerStates ( )

6.562.2.8 bool activemq::state::TransactionState::isPrepared ( ) const [inline]

6.562.2.9 void activemq::state::TransactionState::setPrepared (bool *prepared*) [inline]

6.562.2.10 void activemq::state::TransactionState::setPreparedResult (int *preparedResult*) [inline]

6.562.2.11 void activemq::state::TransactionState::shutdown ( ) [inline]

6.562.2.12 std::string activemq::state::TransactionState::toString ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**TransactionState.h**

6.563 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 >:

6.563.1 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**

6.564 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. 2158).*
- virtual ~**TransferQueue** ()
- virtual void **transfer** (E *e, bool timed, long long nanos)

  Performs a put.
- virtual E * **transfer** (bool timed, long long nanos)

  Performs a take.

6.564.1 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 modes 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).

6.564.2 Constructor & Destructor Documentation

6.564.2.1 template<typename E > decaf::internal::util::concurrent::TransferQueue< E >::TransferQueue ( ) [inline]

Node class for **TransferQueue** (p. 2158).

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.

6.564.2.2 template<typename E > virtual decaf::internal::util::concurrent::TransferQueue< E >::~TransferQueue ( ) [inline, virtual]

6.564.3 Member Function Documentation

6.564.3.1 template<typename E > virtual void decaf::internal::util::concurrent::TransferQueue< E >::transfer (E * *e*, bool *timed*, long long *nanos*) [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. 2158).

6.564.3.2 template<typename E > virtual E* decaf::internal::util::concurrent::TransferQueue< E >::transfer (bool *timed,* long long *nanos*) [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. 2158).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferQueue.h**

6.565 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)

  Performs a put.
- virtual E * **transfer** (bool timed, long long nanos)

  Performs a take.

template<typename E> class decaf::internal::util::concurrent::TransferStack< E >

6.565.1 Constructor & Destructor Documentation

6.565.1.1 template<typename E > decaf::internal::util::concurrent::TransferStack< E >::TransferStack ( ) [inline]

6.565.1.2 template<typename E > virtual decaf::internal::util::concurrent::TransferStack< E >::~TransferStack ( ) [inline, virtual]

6.565.2 Member Function Documentation

6.565.2.1 template<typename E > virtual void decaf::internal::util::concurrent::TransferStack< E >::transfer (E * e, bool *timed*, long long *nanos*) [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. 2158).

6.565.2.2 template<typename E > virtual E* decaf::internal::util::concurrent::TransferStack< E >::transfer (bool *timed*, long long *nanos*) [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. 2158).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferStack.h**

6.566 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

  *Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.*
- virtual void **stop** ()=0

  *Stops the **Transport** (p. 2161).*
- virtual void **oneway** (**const Pointer**< **Command** > &command)=0

  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command)=0

  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

  Sends the given command to the broker and then waits for the response.
- virtual **Pointer**
  < **wireformat::WireFormat** > **getWireFormat** () **const** =0

  Gets the WireFormat instance that is in use by this transport.
- 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. 2161) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () **const** =0

  *Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () **const** =0

  *Is the **Transport** (p. 2161) Connected to its Broker.*
- virtual bool **isClosed** () **const** =0

  *Has the **Transport** (p. 2161) been shutdown and no longer usable.*
- virtual bool **isReconnectSupported** () **const** =0
- virtual bool **isUpdateURIsSupported** () **const** =0
- virtual std::string **getRemoteAddress** () **const** =0
- virtual void **reconnect** (**const decaf::net::URI** &uri)=0

  reconnect to another location
- virtual void **updateURIs** (bool rebalance, **const decaf::util::List**< **decaf::net::URI** > &uris)=0

  *Updates the set of URIs the **Transport** (p. 2161) can connect to.*

6.566.1 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. 2161) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 2161) 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.

6.566.2 Constructor & Destructor Documentation

6.566.2.1 virtual activemq::transport::Transport::~Transport ( ) [inline, virtual]

6.566.3 Member Function Documentation

6.566.3.1 virtual std::string activemq::transport::Transport::getRemoteAddress ( ) const [pure virtual]

Returns

the remote address for this connection

Implemented in **activemq::transport::IOTransport** (p. 1202), **activemq::transport::mock::MockTransport** (p. 1512), **activemq::transport::failover::FailoverTransport** (p. 1013), and **activemq::transport::TransportFilter** (p. 2171).

6.566.3.2 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::IOTransport** (p. 1202), **activemq::transport::mock::MockTransport** (p. 1512), **activemq::transport::failover::FailoverTransport** (p. 1013), and **activemq::transport::TransportFilter** (p. 2171).

6.566.3.3 virtual Pointer<wireformat::WireFormat> activemq::transport::Transport::getWireFormat ( ) const [pure virtual]

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a WireFormat info instance.

Returns

The WireFormat the object used to encode / decode commands.

Implemented in **activemq::transport::IOTransport** (p. 1203), **activemq::transport::mock::MockTransport** (p. 1513), **activemq::transport::failover::FailoverTransport** (p. 1013), and **activemq::transport::TransportFilter** (p. 2171).

6.566.3.4 virtual bool activemq::transport::Transport::isClosed ( ) const [pure virtual]

Has the **Transport** (p. 2161) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2161)

Implemented in **activemq::transport::IOTransport** (p. 1203), **activemq::transport::mock::MockTransport** (p. 1513), **activemq::transport::TransportFilter** (p. 2172), **activemq::transport::failover::FailoverTransport** (p. 1014), and **activemq::transport::tcp::TcpTransport** (p. 2088).

6.566.3.5 virtual bool activemq::transport::Transport::isConnected ( ) const [pure virtual]

Is the **Transport** (p. 2161) Connected to its Broker.

Returns

true if a connection has been made.

Implemented in **activemq::transport::IOTransport** (p. 1203), **activemq::transport::mock::MockTransport** (p. 1513), **activemq::transport::failover::FailoverTransport** (p. 1014), **activemq::transport::TransportFilter** (p. 2172), and **activemq::transport::tcp::TcpTransport** (p. 2088).

6.566.3.6 virtual bool activemq::transport::Transport::isFaultTolerant ( ) const [pure virtual]

Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2161) is fault tolerant.

Implemented in **activemq::transport::IOTransport** (p. 1203), **activemq::transport::mock::MockTransport** (p. 1513), **activemq::transport::failover::FailoverTransport** (p. 1015), **activemq::transport::TransportFilter** (p. 2172), and **activemq::transport::tcp::TcpTransport** (p. 2089).

6.566.3.7 virtual bool activemq::transport::Transport::isReconnectSupported ( ) const [pure virtual]

Returns

true if reconnect is supported.

Implemented in **activemq::transport::mock::MockTransport** (p. 1514), **activemq::transport::failover::FailoverTransport** (p. 1015), **activemq::transport::IOTransport** (p. 1203), and **activemq::transport::TransportFilter** (p. 2172).

6.566.3.8 virtual bool activemq::transport::Transport::isUpdateURIsSupported ( ) const [pure virtual]

Returns

true if updating uris is supported.

Implemented in **activemq::transport::mock::MockTransport** (p. 1514), **activemq::transport::failover::FailoverTransport** (p. 1015), **activemq::transport::IOTransport** (p. 1204), and **activemq::transport::TransportFilter** (p. 2172).

6.566.3.9 virtual Transport* activemq::transport::Transport::narrow (const std::type_info & *typeId*) [pure virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2161) 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::IOTransport** (p. 1204), **activemq::transport::mock::MockTransport** (p. 1514), **activemq::transport::failover::FailoverTransport** (p. 1016), and **activemq::transport::TransportFilter** (p. 2172).

6.566.3.10 virtual void activemq::transport::Transport::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 1514), **activemq::transport::IOTransport** (p. 1204), **activemq::transport::failover::FailoverTransport** (p. 1016), **activemq::transport::TransportFilter** (p. 2173), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1597), **activemq::transport::correlator::ResponseCorrelator** (p. 1787), **activemq::transport::inactivity::InactivityMonitor** (p. 1105), and **activemq::transport::logging::LoggingTransport** (p. 1326).

6.566.3.11 virtual void activemq::transport::Transport::reconnect (const decaf::net::URI & *uri*) [pure virtual]

reconnect to another location

Parameters

uri	The new URI to connect this **Transport** (p. 2161) to.

Exceptions

IOException	on failure or if reconnect is not supported.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1016), and **activemq::transport::TransportFilter** (p. 2174).

6.566.3.12 virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 1515), **activemq::transport::IOTransport** (p. 1204), **activemq::transport::failover::FailoverTransport** (p. 1017), **activemq::transport::TransportFilter** (p. 2174), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1598), **activemq::transport::correlator::ResponseCorrelator** (p. 1787), and **activemq::transport::logging::LoggingTransport** (p. 1326).

6.566.3.13 virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & *command,* unsigned int *timeout*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implemented in **activemq::transport::IOTransport** (p. 1205), **activemq::transport::mock::MockTransport** (p. 1515), **activemq::transport::failover::FailoverTransport** (p. 1017), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1598), **activemq::transport::TransportFilter** (p. 2174), **activemq::transport::correlator::ResponseCorrelator** (p. 1788), and **activemq::transport::logging::LoggingTransport** (p. 1327).

6.566.3.14 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::IOTransport** (p. 1206), **activemq::transport::mock::MockTransport** (p. 1516), **activemq::transport::failover::FailoverTransport** (p. 1018), and **activemq::transport::TransportFilter** (p. 2175).

6.566.3.15 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.

Implemented in **activemq::transport::IOTransport** (p. 1206), and **activemq::transport::TransportFilter** (p. 2175).

6.566.3.16 virtual void activemq::transport::Transport::start ( ) [pure virtual]

Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.

Exceptions

IOException	if and error occurs while starting the **Transport** (p. 2161).

Implemented in **activemq::transport::IOTransport** (p. 1206), **activemq::transport::mock::MockTransport** (p. 1517), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1599), **activemq::transport::TransportFilter** (p. 2175), **activemq::transport::failover::FailoverTransport** (p. 1019), and **activemq::transport::correlator::ResponseCorrelator** (p. 1788).

6.566.3.17 virtual void activemq::transport::Transport::stop ( ) [pure virtual]

Stops the **Transport** (p. 2161).

Exceptions

IOException	if an error occurs while stopping the transport.

Implemented in **activemq::transport::IOTransport** (p. 1206), **activemq::transport::mock::MockTransport** (p. 1517), **activemq::transport::TransportFilter** (p. 2175), and **activemq::transport::failover::FailoverTransport** (p. 1019).

6.566.3.18 virtual void activemq::transport::Transport::updateURIs (bool *rebalance*, const decaf::util::List< decaf::net::URI > & *uris*) [pure virtual]

Updates the set of URIs the **Transport** (p. 2161) can connect to.

If the **Transport** (p. 2161) doesn't support updating its URIs then an IOException is thrown.

Parameters

rebalance	Indicates if a forced reconnection should be performed as a result of the update.
uris	The new list of URIs that can be used for connection.

Exceptions

IOException	if an error occurs or updates aren't supported.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1019), and **activemq::transport::TransportFilter** (p. 2176).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**Transport.h**

6.567 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

  *Creates a fully configured **Transport** (p. 2161) instance which could be a chain of filters and transports.*
- virtual **Pointer**< **Transport** > **createComposite** (**const decaf::net::URI** &location)=0

  *Creates a slimed down **Transport** (p. 2161) instance which can be used in composite transport instances.*

6.567.1 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. 2161) 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

6.567.2 Constructor & Destructor Documentation

6.567.2.1 virtual activemq::transport::TransportFactory::~TransportFactory ( ) [inline, virtual]

6.567.3 Member Function Documentation

6.567.3.1 virtual Pointer<Transport> activemq::transport::TransportFactory::create (const decaf::net::URI & *location*) [pure virtual]

Creates a fully configured **Transport** (p. 2161) 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. 1020), **activemq::transport::mock::MockTransportFactory** (p. 1518), and **activemq::transport::tcp::TcpTransportFactory** (p. 2089).

6.567.3.2 virtual Pointer<Transport> activemq::transport::TransportFactory::createComposite (const decaf::net::URI & *location*) [pure virtual]

Creates a slimed down **Transport** (p. 2161) 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::mock::MockTransportFactory** (p. 1518), **activemq::transport::failover::FailoverTransportFactory** (p. 1021), and **activemq::transport::tcp::TcpTransportFactory** (p. 2090).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportFactory.h**

6.568 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)
  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command)
  Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (**const Pointer**< **Command** > &command, unsigned int timeout)
  Sends the given command to the broker and then waits for the response.
- virtual void **setTransportListener** (**TransportListener** *__**listener**)
  Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () **const**
  Gets the observer of asynchronous events from this transport.

- virtual **Pointer**
  < **wireformat::WireFormat** > **getWireFormat** () **const**

  Gets the WireFormat instance that is in use by this transport.
- virtual void **setWireFormat** (**const Pointer**< **wireformat::WireFormat** > &wireFormat)

  Sets the WireFormat instance to use.
- virtual void **start** ()

  *Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.*
- virtual void **stop** ()

  *Stops the **Transport** (p. 2161).*
- virtual void **close** ()

  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. 2161) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () **const**

  *Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () **const**

  *Is the **Transport** (p. 2161) Connected to its Broker.*
- virtual bool **isReconnectSupported** () **const**
- virtual bool **isUpdateURIsSupported** () **const**
- virtual bool **isClosed** () **const**

  *Has the **Transport** (p. 2161) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () **const**
- virtual void **reconnect** (**const decaf::net::URI** &uri)

  reconnect to another location
- virtual void **updateURIs** (bool rebalance, **const decaf::util::List**< **decaf::net::URI** > &uris)

  *Updates the set of URIs the **Transport** (p. 2161) can connect to.*

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.

6.568.1 Detailed Description

A filter on the transport layer.

Transport (p. 2161) filters implement the **Transport** (p. 2161) interface and optionally delegate calls to another **Transport** (p. 2161) object.

Since

1.0

6.568.2 Constructor & Destructor Documentation

6.568.2.1 activemq::transport::TransportFilter::TransportFilter (const Pointer< Transport > & *next*)

Constructor.

Parameters

next	- the next **Transport** (p. 2161) in the chain

6.568.2.2 virtual activemq::transport::TransportFilter::~TransportFilter ( ) [virtual]

6.568.3 Member Function Documentation

6.568.3.1 virtual void activemq::transport::TransportFilter::close ( ) [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. 633).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1597), **activemq::transport::tcp::TcpTransport** (p. 2087), **activemq::transport::correlator::ResponseCorrelator** (p. 1786), and **activemq::transport::inactivity::InactivityMonitor** (p. 1104).

6.568.3.2 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

6.568.3.3 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

6.568.3.4 virtual std::string activemq::transport::TransportFilter::getRemoteAddress ( ) const [inline, virtual]

Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2162).

6.568.3.5 virtual TransportListener* activemq::transport::TransportFilter::getTransportListener ( ) const [inline, virtual]

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implements **activemq::transport::Transport** (p. 2163).

6.568.3.6 virtual Pointer<wireformat::WireFormat> activemq::transport::TransportFilter::getWireFormat ( ) const [inline, virtual]

Gets the WireFormat instance that is in use by this transport.

In the case of nested transport this method delegates down to the lowest level transport that actually maintains a WireFormat info instance.

Returns

The WireFormat the object used to encode / decode commands.

Implements **activemq::transport::Transport** (p. 2163).

6.568.3.7 virtual bool activemq::transport::TransportFilter::isClosed ( ) const [inline, virtual]

Has the **Transport** (p. 2161) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 2161)

Implements **activemq::transport::Transport** (p. 2163).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2088).

6.568.3.8 virtual bool activemq::transport::TransportFilter::isConnected ( ) const [inline, virtual]

Is the **Transport** (p. 2161) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2163).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2088).

6.568.3.9 virtual bool activemq::transport::TransportFilter::isFaultTolerant ( ) const [inline, virtual]

Is this **Transport** (p. 2161) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 2161) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2163).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2089).

6.568.3.10 virtual bool activemq::transport::TransportFilter::isReconnectSupported ( ) const [inline, virtual]

Returns

true if reconnect is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.568.3.11 virtual bool activemq::transport::TransportFilter::isUpdateURIsSupported ( ) const [inline, virtual]

Returns

true if updating uris is supported.

Implements **activemq::transport::Transport** (p. 2164).

6.568.3.12 virtual Transport* activemq::transport::TransportFilter::narrow (const std::type_info & *typeId*) [virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2161) 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. 2164).

6.568.3.13 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. 2177).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1597), **activemq::transport::correlator::ResponseCorrelator** (p. 1787), **activemq::transport::inactivity::InactivityMonitor** (p. 1105), and **activemq::transport::logging::LoggingTransport** (p. 1326).

6.568.3.14 virtual void activemq::transport::TransportFilter::oneway (const Pointer< Command > & *command*) [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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2164).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1597), **activemq::transport::correlator::ResponseCorrelator** (p. 1787), **activemq::transport::inactivity::InactivityMonitor** (p. 1105), and **activemq::transport::logging::LoggingTransport** (p. 1326).

6.568.3.15 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. 2177).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 1105).

6.568.3.16 virtual void activemq::transport::TransportFilter::reconnect (const decaf::net::URI & *uri*) [virtual]

reconnect to another location

Parameters

uri	The new URI to connect this **Transport** (p. 2161) to.

Exceptions

IOException	on failure or if reconnect is not supported.

Implements **activemq::transport::Transport** (p. 2165).

6.568.3.17 virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & *command*) [inline, 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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2165).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1598), **activemq::transport::correlator::ResponseCorrelator** (p. 1787), and **activemq::transport::logging::LoggingTransport** (p. 1326).

6.568.3.18 virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & *command,* unsigned int *timeout*) [inline, 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.
UnsupportedOperation-Exception	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 2165).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1598), **activemq::transport::correlator::ResponseCorrelator** (p. 1788), and **activemq::transport::logging::LoggingTransport** (p. 1327).

6.568.3.19 virtual void activemq::transport::TransportFilter::setTransportListener (TransportListener * *listener*) [inline, virtual]

Sets the observer of asynchronous events from this transport.

Parameters

listener	the listener of transport events.

Implements **activemq::transport::Transport** (p. 2166).

6.568.3.20 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.

Implements **activemq::transport::Transport** (p. 2166).

6.568.3.21 virtual void activemq::transport::TransportFilter::start ( ) [virtual]

Starts the **Transport** (p. 2161), the send methods of a **Transport** (p. 2161) will throw an exception if used before the **Transport** (p. 2161) is started.

Exceptions

IOException	if and error occurs while starting the **Transport** (p. 2161).

Implements **activemq::transport::Transport** (p. 2166).

Reimplemented in **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1599), and **activemq::transport::correlator::ResponseCorrelator** (p. 1788).

6.568.3.22 virtual void activemq::transport::TransportFilter::stop ( ) [virtual]

Stops the **Transport** (p. 2161).

Exceptions

IOException	if an error occurs while stopping the transport.

Implements **activemq::transport::Transport** (p. 2167).

6.568.3.23 virtual void activemq::transport::TransportFilter::transportInterrupted ( ) [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2178).

6.568.3.24 virtual void activemq::transport::TransportFilter::transportResumed ( ) [virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2178).

6.568.3.25 virtual void activemq::transport::TransportFilter::updateURIs (bool *rebalance*, const decaf::util::List< decaf::net::URI > & *uris*) [inline, virtual]

Updates the set of URIs the **Transport** (p. 2161) can connect to.

If the **Transport** (p. 2161) doesn't support updating its URIs then an IOException is thrown.

Parameters

rebalance	Indicates if a forced reconnection should be performed as a result of the update.
uris	The new list of URIs that can be used for connection.

Exceptions

IOException	if an error occurs or updates aren't supported.

Implements **activemq::transport::Transport** (p. 2167).

6.568.4 Field Documentation

6.568.4.1 TransportListener* activemq::transport::TransportFilter::listener [protected]

Listener of this transport.

6.568.4.2 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**

6.569 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.

6.569.1 Detailed Description

A listener of asynchronous exceptions from a command transport object.

6.569.2 Constructor & Destructor Documentation

6.569.2.1 virtual activemq::transport::TransportListener::~TransportListener ( ) [inline, virtual]

6.569.3 Member Function Documentation

6.569.3.1 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. 2161) deletes the command upon receipt.

Parameters

command	The received command object.

Implemented in **activemq::core::ActiveMQConnection** (p. 159), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1597), **activemq::transport::correlator::ResponseCorrelator** (p. 1787), **activemq::transport::TransportFilter** (p. 2173), **activemq::transport::inactivity::InactivityMonitor** (p. 1105), **activemq::transport::mock::InternalCommandListener** (p. 1184), **activemq::transport::failover::FailoverTransportListener** (p. 1022), and **activemq::transport::logging::LoggingTransport** (p. 1326).

6.569.3.2 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. 159), **activemq::transport::failover::BackupTransport** (p. 378), **activemq::transport::TransportFilter** (p. 2173), **activemq::transport::inactivity::InactivityMonitor** (p. 1105), and **activemq::transport::failover::FailoverTransportListener** (p. 1022).

6.569.3.3 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. 165), **activemq::transport::TransportFilter** (p. 2176), **activemq::transport::failover::FailoverTransportListener** (p. 1023), and **activemq::transport::DefaultTransportListener** (p. 913).

6.569.3.4 virtual void activemq::transport::TransportListener::transportResumed ( ) [pure virtual]

The transport has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnection** (p. 165), **activemq::transport::TransportFilter** (p. 2176), **activemq::transport::failover::FailoverTransportListener** (p. 1023), and **activemq::transport::DefaultTransportListener** (p. 913).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportListener.h**

6.570 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 2161) 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**

  *Gets a Registered **TransportFactory** (p. 2167) 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)

*Registers a new **TransportFactory** (p. 2167) with this Registry.*

- void **unregisterFactory** (**const** std::string &name)

  Unregisters the Factory with the given name and deletes that instance of the Factory.

- void **unregisterAllFactories** ()

  Removes all Factories and deletes the instances of the Factory objects.

- std::vector< std::string > **getTransportNames** () **const**

  *Retrieves a list of the names of all the Registered **Transport** (p. 2161)'s in this Registry.*

Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()

  *Gets the single instance of the **TransportRegistry** (p. 2178).*

6.570.1 Detailed Description

Registry of all **Transport** (p. 2161) Factories that are available to the client at runtime.

New **Transport** (p. 2161)'s must have a factory registered here before a connection attempt is made.

Since

3.0

6.570.2 Constructor & Destructor Documentation

6.570.2.1 virtual activemq::transport::TransportRegistry::~TransportRegistry ( ) [virtual]

6.570.3 Member Function Documentation

6.570.3.1 TransportFactory* activemq::transport::TransportRegistry::findFactory (const std::string & *name*) const

Gets a Registered **TransportFactory** (p. 2167) 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.

6.570.3.2 static TransportRegistry& activemq::transport::TransportRegistry::getInstance ( ) [static]

Gets the single instance of the **TransportRegistry** (p. 2178).

Returns

reference to the single instance of this Registry

6.570.3.3 std::vector<std::string> activemq::transport::TransportRegistry::getTransportNames ( ) const

Retrieves a list of the names of all the Registered **Transport** (p. 2161)'s in this Registry.

Returns

stl vector of strings with all the **Transport** (p. 2161) names registered.

6.570.3.4 void activemq::transport::TransportRegistry::registerFactory (const std::string & *name,* TransportFactory * *factory*)

Registers a new **TransportFactory** (p. 2167) 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.

6.570.3.5 void activemq::transport::TransportRegistry::unregisterAllFactories ( )

Removes all Factories and deletes the instances of the Factory objects.

6.570.3.6 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**

6.571 tree_desc_s Struct Reference

`#include <src/main/decaf/internal/util/zip/deflate.h>`

Data Fields

- **ct_data** * **dyn_tree**

- int **max_code**
- **static_tree_desc** * **stat_desc**

6.571.1 Field Documentation

6.571.1.1 ct_data* tree_desc_s::dyn_tree

6.571.1.2 int tree_desc_s::max_code

6.571.1.3 static_tree_desc* tree_desc_s::stat_desc

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**deflate.h**

6.572 decaf::lang::Thread::UncaughtExceptionHandler Class Reference

Interface for handlers invoked when a **Thread** (p. 2094) abruptly terminates due to an uncaught exception.

`#include <src/main/decaf/lang/Thread.h>`

Public Member Functions

- virtual ~**UncaughtExceptionHandler** ()
- virtual void **uncaughtException** (**const Thread** *thread, **const Throwable** &error)=0 throw ()

  Method invoked when the given thread terminates due to the given uncaught exception.

6.572.1 Detailed Description

Interface for handlers invoked when a **Thread** (p. 2094) abruptly terminates due to an uncaught exception.

6.572.2 Constructor & Destructor Documentation

6.572.2.1 virtual decaf::lang::Thread::UncaughtExceptionHandler::~UncaughtExceptionHandler ( ) [inline, virtual]

6.572.3 Member Function Documentation

6.572.3.1 virtual void decaf::lang::Thread::UncaughtExceptionHandler::uncaughtException (const Thread * *thread,* const Throwable & *error*) throw () [pure virtual]

Method invoked when the given thread terminates due to the given uncaught exception.

This method is defined to indicate that it will not throw an exception, throwing and exception from this method will on most systems result in a segmentation fault.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Thread.h**

6.573 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 ()

6.573.1 Constructor & Destructor Documentation

6.573.1.1 decaf::net::UnknownHostException::UnknownHostException ( ) throw () [inline]

Default Constructor.

6.573.1.2 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

6.573.1.3 decaf::net::UnknownHostException::UnknownHostException (const UnknownHostException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.573.1.4 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

6.573.1.5 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.

6.573.1.6 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

6.573.1.7 virtual decaf::net::UnknownHostException::~UnknownHostException ( ) throw () [inline, virtual]

6.573.2 Member Function Documentation

6.573.2.1 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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownHostException.h**

6.574 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 ()

6.574.1 Constructor & Destructor Documentation

6.574.1.1 decaf::net::UnknownServiceException::UnknownServiceException ( ) throw () [inline]

Default Constructor.

6.574.1.2 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

6.574.1.3 decaf::net::UnknownServiceException::UnknownServiceException (const UnknownServiceException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.574.1.4 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

6.574.1.5 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.

6.574.1.6 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

6.574.1.7 virtual decaf::net::UnknownServiceException::~UnknownServiceException ( ) throw () [inline, virtual]

6.574.2 Member Function Documentation

6.574.2.1 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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownServiceException.h**

6.575 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 ()

6.575.1 Detailed Description

Thrown when the the Character Encoding is not supported.

Since

1.0

6.575.2 Constructor & Destructor Documentation

6.575.2.1 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( ) throw () [inline]

Default Constructor.

6.575.2.2 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.575.2.3 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const UnsupportedEncodingException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.575.2.4 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

6.575.2.5 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.

6.575.2.6 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

6.575.2.7 virtual decaf::io::UnsupportedEncodingException::~UnsupportedEncodingException ( ) throw () [inline, virtual]

6.575.3 Member Function Documentation

6.575.3.1 virtual UnsupportedEncodingException* decaf::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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**UnsupportedEncodingException.h**

6.576 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. 990).*
- **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 ()

6.576.1 Constructor & Destructor Documentation

6.576.1.1 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException ( ) throw () [inline]

Default Constructor.

6.576.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 990).

Parameters

ex	An exception that should become this type of **Exception** (p. 990)

6.576.1.3 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException (const UnsupportedOperationException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of **Exception** (p. 990)

6.576.1.4 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

6.576.1.5 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause	**Pointer** (p. 1614) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.576.1.6 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

6.576.1.7 virtual decaf::lang::exceptions::UnsupportedOperationException::~UnsupportedOperationException ( ) throw () [inline, virtual]

6.576.2 Member Function Documentation

6.576.2.1 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. 990) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 992).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 1741).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**UnsupportedOperationException.h**

6.577 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** ()
- **UnsupportedOperationException** (**const UnsupportedOperationException** &ex)
- **UnsupportedOperationException** (**const** std::string &message)
- **UnsupportedOperationException** (**const** std::string &message, **const** std::exception *cause)
- **UnsupportedOperationException** (**const** std::string &message, **const** std::exception *cause, **const** std::vector< std::pair< std::string, int > > &stackTrace)
- virtual ~**UnsupportedOperationException** () throw ()

6.577.1 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

6.577.2 Constructor & Destructor Documentation

6.577.2.1 cms::UnsupportedOperationException::UnsupportedOperationException ( )

6.577.2.2 cms::UnsupportedOperationException::UnsupportedOperationException (const UnsupportedOperationException & *ex*)

6.577.2.3 cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & *message*)

6.577.2.4 cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & *message,* const std::exception * *cause*)

6.577.2.5 cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & *message,* const std::exception * *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.577.2.6 virtual cms::UnsupportedOperationException::~UnsupportedOperationException ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**UnsupportedOperationException.h**

6.578 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 2191) 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)

  *Constructs a **URI** (p. 2191) as a copy of another **URI** (p. 2191).*
- **URI** (**const** std::string &uri)

  *Constructs a **URI** (p. 2191) from the given string.*
- **URI** (**const** std::string &scheme, **const** std::string &ssp, **const** std::string &fragment)

  *Constructs a **URI** (p. 2191) 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)

*Constructs a **URI** (p. 2191) from the given components.*

- **URI** (**const** std::string &scheme, **const** std::string &host, **const** std::string &path, **const** std::string &fragment)

  *Constructs a **URI** (p. 2191) 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)

  *Constructs a **URI** (p. 2191) 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. 2191).*
- std::string **getRawFragment** () **const**

  *Returns the raw fragment component of this **URI** (p. 2191).*
- std::string **getRawPath** () **const**

  *Returns the raw path component of this **URI** (p. 2191).*
- std::string **getRawQuery** () **const**

  *Returns the raw query component of this **URI** (p. 2191).*
- std::string **getRawSchemeSpecificPart** () **const**

  *Returns the raw scheme-specific part of this **URI** (p. 2191).*
- std::string **getSchemeSpecificPart** () **const**

  *Returns the decoded scheme-specific part of this **URI** (p. 2191).*
- std::string **getRawUserInfo** () **const**

  *Returns the raw user-information component of this **URI** (p. 2191).*
- bool **isAbsolute** () **const**

  *Tells whether or not this **URI** (p. 2191) is absolute.*
- bool **isOpaque** () **const**

  *Tells whether or not this **URI** (p. 2191) is opaque.*
- **URI normalize** () **const**

  *Normalizes this **URI** (p. 2191)'s path.*
- **URI parseServerAuthority** () **const**

  *Attempts to parse this **URI** (p. 2191)'s authority component, if defined, into user-information, host, and port components.*
- **URI relativize** (**const URI** &uri) **const**

  *Relativizes the given **URI** (p. 2191) against this **URI** (p. 2191).*
- **URI resolve** (**const** std::string &str) **const**

  *Constructs a new **URI** (p. 2191) by parsing the given string and then resolving it against this **URI** (p. 2191).*
- **URI resolve** (**const URI** &uri) **const**

  *Resolves the given **URI** (p. 2191) against this **URI** (p. 2191).*

- std::string **toString** () **const**

  *Returns the content of this **URI** (p. 2191) as a string.*
- **URL toURL** () **const**

  *Constructs a **URL** (p. 2223) from this **URI** (p. 2191).*

Static Public Member Functions

- static **URI create** (**const** std::string uri)

  *Creates a **URI** (p. 2191) by parsing the given string.*

6.578.1 Detailed Description

This class represents an instance of a **URI** (p. 2191) as defined by RFC 2396.

6.578.2 Constructor & Destructor Documentation

6.578.2.1 decaf::net::URI::URI ( )

Default Constructor, same as calling a Constructor with all fields empty.

6.578.2.2 decaf::net::URI::URI (const URI & *uri*)

Constructs a **URI** (p. 2191) as a copy of another **URI** (p. 2191).

Parameters

uri	- uri to copy

Exceptions

URISyntaxException *(p. 2214)*	if the **URI** (p. 2191) passed is malformed.

6.578.2.3 decaf::net::URI::URI (const std::string & *uri*)

Constructs a **URI** (p. 2191) from the given string.

Parameters

uri	- string uri to parse.

Exceptions

URISyntaxException *(p. 2214)*	if the **URI** (p. 2191) passed is malformed.

6.578.2.4 decaf::net::URI::URI (const std::string & *scheme,* const std::string & *ssp,* const std::string & *fragment*)

Constructs a **URI** (p. 2191) from the given components.

Parameters

scheme	- the uri scheme
ssp	- Scheme specific part
fragment	- Fragment

Exceptions

URISyntaxException *(p. 2214)*	if the **URI** (p. 2191) passed is malformed.

6.578.2.5 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*)

Constructs a **URI** (p. 2191) 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

Exceptions

URISyntaxException *(p. 2214)*	if the **URI** (p. 2191) passed is malformed.

6.578.2.6 decaf::net::URI::URI (const std::string & *scheme,* const std::string & *host,* const std::string & *path,* const std::string & *fragment*)

Constructs a **URI** (p. 2191) from the given components.

Parameters

scheme	- Scheme name
host	- Host name
path	- Path
fragment	- Fragment

Exceptions

URISyntaxException *(p. 2214)*	if the **URI** (p. 2191) passed is malformed.

6.578.2.7 decaf::net::URI::URI (const std::string & *scheme,* const std::string & *authority,* const std::string & *path,* const std::string & *query,* const std::string & *fragment*)

Constructs a **URI** (p. 2191) from the given components.

Parameters

scheme	- Scheme name
authority	- Authority
path	- Path
query	- Query
fragment	- Fragment

Exceptions

URISyntaxException *(p. 2214)*	if the **URI** (p. 2191) passed is malformed.

6.578.2.8 virtual decaf::net::URI::~URI ( ) [inline, virtual]

6.578.3 Member Function Documentation

6.578.3.1 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.

6.578.3.2 static URI decaf::net::URI::create (const std::string *uri*) [static]

Creates a **URI** (p. 2191) by parsing the given string.

This convenience factory method works as if by invoking the URI(string) constructor; any **URISyntaxException** (p. 2214) thrown by the constructor is caught and wrapped in a new IllegalArgumentException object, which is then thrown.

Parameters

uri	- **URI** (p. 2191) string to parse

Exceptions

IllegalArgumentException	

6.578.3.3 virtual bool decaf::net::URI::equals (const URI & *value*) const [virtual]

Returns

true if this value is considered equal to the passed value.

6.578.3.4 std::string decaf::net::URI::getAuthority ( ) const

Returns

the decoded authority component of this **URI** (p. 2191).

6.578.3.5 std::string decaf::net::URI::getFragment ( ) const

Returns

the decoded fragment component of this **URI** (p. 2191).

6.578.3.6 std::string decaf::net::URI::getHost ( ) const

Returns

the host component of this **URI** (p. 2191).

6.578.3.7 std::string decaf::net::URI::getPath ( ) const

Returns

the path component of this **URI** (p. 2191).

6.578.3.8 int decaf::net::URI::getPort ( ) const

Returns

the port component of this **URI** (p. 2191).

6.578.3.9 std::string decaf::net::URI::getQuery ( ) const

Returns

the query component of this **URI** (p. 2191).

6.578.3.10 std::string decaf::net::URI::getRawAuthority ( ) const

Returns the raw authority component of this **URI** (p. 2191).

The authority component of a **URI** (p. 2191), 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. 2191)

6.578.3.11 std::string decaf::net::URI::getRawFragment ( ) const

Returns the raw fragment component of this **URI** (p. 2191).

The fragment component of a **URI** (p. 2191), if defined, only contains legal **URI** (p. 2191) characters.

Returns

the raw fragment component of this **URI** (p. 2191)

6.578.3.12 std::string decaf::net::URI::getRawPath ( ) const

Returns the raw path component of this **URI** (p. 2191).

The path component of a **URI** (p. 2191), 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. 2191)

6.578.3.13 std::string decaf::net::URI::getRawQuery ( ) const

Returns the raw query component of this **URI** (p. 2191).

The query component of a **URI** (p. 2191), if defined, only contains legal **URI** (p. 2191) characters.

Returns

the raw query component of the **URI** (p. 2191).

6.578.3.14 std::string decaf::net::URI::getRawSchemeSpecificPart ( ) const

Returns the raw scheme-specific part of this **URI** (p. 2191).

The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 2191) only contains legal **URI** (p. 2191) characters.

Returns

the raw scheme special part of the uri

6.578.3.15 std::string decaf::net::URI::getRawUserInfo ( ) const

Returns the raw user-information component of this **URI** (p. 2191).

The user-information component of a **URI** (p. 2191), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

Returns

the raw user-information component of the **URI** (p. 2191)

6.578.3.16 std::string decaf::net::URI::getScheme ( ) const

Returns

the scheme component of this **URI** (p. 2191)

6.578.3.17 std::string decaf::net::URI::getSchemeSpecificPart ( ) const

Returns the decoded scheme-specific part of this **URI** (p. 2191).

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.

6.578.3.18 std::string decaf::net::URI::getUserInfo ( ) const

Returns

the user info component of this **URI** (p. 2191)

6.578.3.19 bool decaf::net::URI::isAbsolute ( ) const

Tells whether or not this **URI** (p. 2191) is absolute.

A **URI** (p. 2191) is absolute if, and only if, it has a scheme component.

Returns

true if, and only if, this **URI** (p. 2191) is absolute

6.578.3.20 bool decaf::net::URI::isOpaque ( ) const

Tells whether or not this **URI** (p. 2191) is opaque.

A **URI** (p. 2191) 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. 2191) 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. 2191) is opaque

6.578.3.21 URI decaf::net::URI::normalize ( ) const

Normalizes this **URI** (p. 2191)'s path.

If this **URI** (p. 2191) is opaque, or if its path is already in normal form, then this **URI** (p. 2191) is returned. Otherwise a new **URI** (p. 2191) is constructed that is identical to this **URI** (p. 2191) except that its path is computed by normalizing this **URI** (p. 2191)'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.
   (a) 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.
   (b) 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. 2191) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 2191) 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. 2191) equivalent to this **URI** (p. 2191), but whose path is in normal form

6.578.3.22 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.

6.578.3.23 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.

6.578.3.24 URI decaf::net::URI::parseServerAuthority ( ) const

Attempts to parse this **URI** (p. 2191)'s authority component, if defined, into user-information, host, and port components.

If this **URI** (p. 2191)'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. 2191) has no authority component, this method simply returns this **URI** (p. 2191).

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. 2191) whose authority field has been parsed as a server-based authority

Exceptions

URISyntaxException *(p. 2214)*	- If the authority component of this **URI** (p. 2191) is defined but cannot be parsed as a server-based authority.

6.578.3.25 URI decaf::net::URI::relativize (const URI & *uri*) const

Relativizes the given **URI** (p. 2191) against this **URI** (p. 2191).

The relativization of the given **URI** (p. 2191) against this **URI** (p. 2191) is computed as follows:

1. If either this **URI** (p. 2191) or the given **URI** (p. 2191) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 2191) is not a prefix of the path of the given **URI** (p. 2191), then the given **URI** (p. 2191) is returned.
   (a) Otherwise a new relative hierarchical **URI** (p. 2191) is constructed with query and fragment components taken from the given **URI** (p. 2191) and with a path component computed by removing this **URI** (p. 2191)'s path from the beginning of the given **URI** (p. 2191)'s path.

Parameters

uri	- The **URI** (p. 2191) to be relativized against this **URI** (p. 2191)

Returns

The resulting **URI** (p. 2191)

6.578.3.26 URI decaf::net::URI::resolve (const std::string & *str*) const

Constructs a new **URI** (p. 2191) by parsing the given string and then resolving it against this **URI** (p. 2191).

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. 2191)

Returns

The resulting **URI** (p. 2191)

Exceptions

IllegalArgumentException	- If the given string violates RFC 2396

6.578.3.27 URI decaf::net::URI::resolve (const URI & *uri*) const

Resolves the given **URI** (p. 2191) against this **URI** (p. 2191).

If the given **URI** (p. 2191) is already absolute, or if this **URI** (p. 2191) is opaque, then a copy of the given **URI** (p. 2191) is returned.

If the given **URI** (p. 2191)'s fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 2191) with the given fragment but with all other components equal to those of this **URI** (p. 2191) is returned. This allows a **URI** (p. 2191) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 2191).

Otherwise this method constructs a new hierarchical **URI** (p. 2191) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 2191) is constructed with this **URI** (p. 2191)'s scheme and the given **URI** (p. 2191)'s query and

fragment components.

(a) If the given **URI** (p. 2191) has an authority component then the new **URI** (p. 2191)'s authority and path are taken from the given **URI** (p. 2191).

(b) Otherwise the new **URI** (p. 2191)'s authority component is copied from this **URI** (p. 2191), and its path is computed as follows:

i. If the given **URI** (p. 2191)'s path is absolute then the new **URI** (p. 2191)'s path is taken from the given **URI** (p. 2191).

A. Otherwise the given **URI** (p. 2191)'s path is relative, and so the new **URI** (p. 2191)'s path is computed by resolving the path of the given **URI** (p. 2191) against the path of this **URI** (p. 2191). This is done by concatenating all but the last segment of this **URI** (p. 2191)'s path, if any, with the given **URI** (p. 2191)'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. 2191) is absolute or the given **URI** (p. 2191) is absolute.

Parameters

uri	- The **URI** (p. 2191) to be resolved against this **URI** (p. 2191)

Returns

The resulting **URI** (p. 2191)

6.578.3.28 std::string decaf::net::URI::toString ( ) const

Returns the content of this **URI** (p. 2191) as a string.

If this **URI** (p. 2191) 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. 2191) was created by normalization, resolution, or relativization, and so a string is constructed from this **URI** (p. 2191)'s components according to the rules specified in RFC 2396, section 5.2, step 7.

Returns

the string form of this **URI** (p. 2191)

6.578.3.29 URL decaf::net::URI::toURL ( ) const

Constructs a **URL** (p. 2223) from this **URI** (p. 2191).

This convenience method works as if invoking it were equivalent to evaluating the expression new **URL** (p. 2223)(this.toString()) after first checking that this **URI** (p. 2191) is absolute.

Returns

A **URL** (p. 2223) constructed from this **URI** (p. 2191)

Exceptions

IllegalArgumentException	- If this **URL** (p. 2223) is not absolute
MalformedURLException *(p. 1369)*	- If a protocol handler for the **URL** (p. 2223) could not be found, or if some other error occurred while constructing the **URL** (p. 2223)

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URI.h**

6.579 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)

  Validate a string by checking if it contains any characters other than:
- static void **validateSimple** (**const** std::string &s, **const** std::string &legal)

  Validate a string by checking if it contains any characters other than:
- static std::string **quotelllegal** (**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 hexidecimal 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.

6.579.1 Constructor & Destructor Documentation

6.579.1.1 decaf::internal::net::URIEncoderDecoder::URIEncoderDecoder ( )

6.579.1.2 virtual decaf::internal::net::URIEncoderDecoder::~URIEncoderDecoder ( ) [inline, virtual]

6.579.2 Member Function Documentation

6.579.2.1 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.

6.579.2.2 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 hexidecimal value prepended by ”.

For example: Euro currency symbol -> "%E2%82%AC".

Parameters

s	- the string to be converted

Returns

the converted string

6.579.2.3 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 hexidecimal 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

6.579.2.4 static void decaf::internal::net::URIEncoderDecoder::validate (const std::string & *s,* const std::string & *legal*) [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

Exceptions

URISyntaxException	if the uri string is not well formed.

6.579.2.5 static void decaf::internal::net::URIEncoderDecoder::validateSimple (const std::string & *s*, const std::string & *legal*) [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

Exceptions

URISyntaxException	if the uri string is not well formed.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIEncoderDecoder.h**

6.580 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. 2204) 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)

  Parse the passed in URI.
- void **validateScheme** (**const** std::string &uri, **const** std::string &scheme, int index)

  Validate the schema portin of the URI.
- void **validateSsp** (**const** std::string &uri, **const** std::string &ssp, std::size_t index)

  Validate that the URI Ssp Segment contains no invalid encodings.
- void **validateAuthority** (**const** std::string &uri, **const** std::string &authority, std::size_t index)

  Validate that the URI Authority Segment contains no invalid encodings.
- void **validatePath** (**const** std::string &uri, **const** std::string &path, std::size_t index)

  Validate that the URI Path Segment contains no invalid encodings.

- void **validateQuery** (**const** std::string &uri, **const** std::string &query, std::size_t index)

  Validate that the URI Query Segment contains no invalid encodings.
- void **validateFragment** (**const** std::string &uri, **const** std::string &fragment, std::size_t index)

  Validate that the URI fragment contains no invalid encodings.
- **URIType parseAuthority** (bool forceServer, **const** std::string &authority)

  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)

  Check the supplied user info for validity.
- bool **isValidHost** (bool forceServer, **const** std::string &host)

  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 **isValidIP6Address** (**const** std::string &ipAddress)

  Determines if the given address is valid according to the IPv6 spec.
- bool **isValidIP4Word** (**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.

6.580.1 Detailed Description

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

6.580.2 Constructor & Destructor Documentation

6.580.2.1 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. 2204) 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.

6.580.2.2 decaf::internal::net::URIHelper::URIHelper ( )

Sets up the filter strings with sane defaults.

6.580.2.3 virtual decaf::internal::net::URIHelper::~URIHelper ( ) [inline, virtual]

6.580.3 Member Function Documentation

6.580.3.1 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.

6.580.3.2 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.

6.580.3.3 bool decaf::internal::net::URIHelper::isValidHost (bool *forceServer,* const std::string & *host*)

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 if a valid domain name.

Exceptions

URISyntaxException	if the host is invalid and forceServer is true.

6.580.3.4 bool decaf::internal::net::URIHelper::isValidIP4Word (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.

Parameters

word	- string value to check.

Returns

true if the word is a valid IPv4 word.

6.580.3.5 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.

6.580.3.6 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 were 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.

6.580.3.7 URIType decaf::internal::net::URIHelper::parseAuthority (bool *forceServer,* const std::string & *authority*)

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. 2217) instance containing the parsed data.

Exceptions

URISyntaxException	

6.580.3.8 URIType decaf::internal::net::URIHelper::parseURI (const std::string & *uri,* bool *forceServer*)

Parse the passed in URI.

Parameters

uri	- the URI to Parse
forceServer	- if true invalid URI data throws an Exception

Returns

a **URIType** (p. 2217) instance containing the parsed data.

Exceptions

URISyntaxException	if forceServer is true and the URI is invalid.

6.580.3.9 void decaf::internal::net::URIHelper::validateAuthority (const std::string & *uri,* const std::string & *authority,* std::size_t *index*)

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.

6.580.3.10 void decaf::internal::net::URIHelper::validateFragment (const std::string & *uri,* const std::string & *fragment,* std::size_t *index*)

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.

6.580.3.11 void decaf::internal::net::URIHelper::validatePath (const std::string & *uri,* const std::string & *path,* std::size_t *index*)

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.

6.580.3.12 void decaf::internal::net::URIHelper::validateQuery (const std::string & *uri,* const std::string & *query,* std::size_t *index*)

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.

6.580.3.13 void decaf::internal::net::URIHelper::validateScheme (const std::string & *uri,* const std::string & *scheme,* int *index*)

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.

6.580.3.14 void decaf::internal::net::URIHelper::validateSsp (const std::string & *uri,* const std::string & *ssp,* std::size_t *index*)

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.

6.580.3.15 void decaf::internal::net::URIHelper::validateUserinfo (const std::string & *uri,* const std::string & *userinfo,* std::size_t *index*)

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**

6.581 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.
- ~**URIPool** ()
- **URI getURI** ()

  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 LinkedList**< **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.

6.581.1 Constructor & Destructor Documentation

6.581.1.1 activemq::transport::failover::URIPool::URIPool ( )

Create an Empty URI Pool.

6.581.1.2 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.

6.581.1.3 activemq::transport::failover::URIPool::~URIPool ( )

6.581.2 Member Function Documentation

6.581.2.1 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.

6.581.2.2 void activemq::transport::failover::URIPool::addURIs (const LinkedList< 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.

6.581.2.3 URI activemq::transport::failover::URIPool::getURI ( )

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.

6.581.2.4 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.

6.581.2.5 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

6.581.2.6 void activemq::transport::failover::URIPool::setRandomize (bool *value*) [inline]

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.

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**

6.582 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)

  Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.
- static **CompositeData parseComposite** (**const URI** &uri)

  *Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme:///(uri1,uri2,...uri-N)?param1=value1, each of the composite URIs is stored in the **CompositeData** (p. 690)'s internal list.*
- static **decaf::util::Properties parseQuery** (std::string query)

  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)

  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)

  Given a properties object create a string that can be appended to a URI as a valid Query string.

6.582.1 Member Function Documentation

6.582.1.1 static std::string activemq::util::URISupport::createQueryString (const Properties & *options*) [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.

6.582.1.2 static CompositeData activemq::util::URISupport::parseComposite (const URI & *uri*) [static]

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uri-N)?param1=value1, each of the composite URIs is stored in the **CompositeData** (p. 690)'s internal list.

Parameters

uri	- The Composite URI to parse.

Returns

a new **CompositeData** (p. 690) object with the parsed data

Exceptions

URISyntaxException	if the URI is not well formed.

6.582.1.3 static decaf::util::Properties activemq::util::URISupport::parseQuery (std::string *query*) [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.

6.582.1.4 static void activemq::util::URISupport::parseQuery (std::string *query,* decaf::util::Properties * *properties*) [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.

6.582.1.5 static void activemq::util::URISupport::parseURL (const std::string & *URI,* decaf::util::Properties & *properties*) [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**

6.583 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, int 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**
- int **getIndex** () **const**

6.583.1 Constructor & Destructor Documentation

6.583.1.1 decaf::net::URISyntaxException::URISyntaxException ( ) throw () [inline]

Default Constructor.

6.583.1.2 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

6.583.1.3 decaf::net::URISyntaxException::URISyntaxException (const URISyntaxException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	An exception that should become this type of Exception

6.583.1.4 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

6.583.1.5 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.

6.583.1.6 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

6.583.1.7 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. 2223) that caused the exception.
reason	The reason for the failure.

6.583.1.8 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const std::string & *input*, const std::string & *reason*, int *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. 2191) that caused the exception
reason	The reason for the failure.
index	The index in the **URI** (p. 2191) string where the error occurred.

6.583.1.9 virtual decaf::net::URISyntaxException::~URISyntaxException ( ) throw () [inline, virtual]

6.583.2 Member Function Documentation

6.583.2.1 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. 992).

6.583.2.2 int decaf::net::URISyntaxException::getIndex ( ) const [inline]

Returns

the index in the input string where the error occured or -1

6.583.2.3 std::string decaf::net::URISyntaxException::getInput ( ) const [inline]

Returns

the Input string that cause this exception or ""

6.583.2.4 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**

6.584 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. *2217) instance and the resulting data,.*
- void **setSource** (**const** std::string &source)

  *Sets the source URI string that was parsed to obtain this **URIType*** (p. *2217) 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.

6.584.1 Detailed Description

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

6.584.2 Constructor & Destructor Documentation

6.584.2.1 decaf::internal::net::URIType::URIType (const std::string & *source*) [inline]

6.584.2.2 decaf::internal::net::URIType::URIType ( ) [inline]

6.584.2.3 virtual decaf::internal::net::URIType::~URIType ( ) [inline, virtual]

6.584.3 Member Function Documentation

6.584.3.1 std::string decaf::internal::net::URIType::getAuthority ( ) const [inline]

Gets the Authority of the URI.

Returns

Authority part string.

6.584.3.2 std::string decaf::internal::net::URIType::getFragment ( ) const [inline]

Gets the Fragment part of the URI.

Returns

Fragment part string.

6.584.3.3 std::string decaf::internal::net::URIType::getHost ( ) const [inline]

Gets the Host name part of the URI.

Returns

Host name part string.

6.584.3.4 std::string decaf::internal::net::URIType::getPath ( ) const [inline]

Gets the Path part of the URI.

Returns

Path part string.

6.584.3.5 int decaf::internal::net::URIType::getPort ( ) const [inline]

Gets the port part of the URI.

Returns

port part string, -1 if not set.

6.584.3.6 std::string decaf::internal::net::URIType::getQuery ( ) const [inline]

Gets the Query part of the URI.

Returns

Query part string.

6.584.3.7 std::string decaf::internal::net::URIType::getScheme ( ) const [inline]

Gets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

Returns

scheme part string.

6.584.3.8 std::string decaf::internal::net::URIType::getSchemeSpecificPart ( ) const [inline]

Gets the Scheme Specific Part of the URI.

Returns

scheme specific part string.

6.584.3.9 std::string decaf::internal::net::URIType::getSource ( ) const [inline]

Gets the source URI string that was parsed to obtain this **URIType** (p. 2217) instance and the resulting data,.

Returns

the source URI string

6.584.3.10 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.

6.584.3.11 bool decaf::internal::net::URIType::isAbsolute ( ) const [inline]

Gets if the URI is Absolute.

Returns

true if Absolute.

6.584.3.12 bool decaf::internal::net::URIType::isOpaque ( ) const [inline]

Gets if the URI is Opaque.

Returns

true if opaque.

6.584.3.13 bool decaf::internal::net::URIType::isServerAuthority ( ) const [inline]

Gets if the URI is a Server Authority.

Returns

true if Server Authority.

6.584.3.14 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. 2217) contains valid data.

6.584.3.15 void decaf::internal::net::URIType::setAbsolute (bool *absolute*) [inline]

Sets if the URI is Absolute.

Parameters

absolute	- true if Absolute.

6.584.3.16 void decaf::internal::net::URIType::setAuthority (const std::string & *authority*) [inline]

Sets the Authority of the URI.

Parameters

authority	Authority part string.

6.584.3.17 void decaf::internal::net::URIType::setFragment (const std::string & *fragment*) [inline]

Sets the Fragment part of the URI.

Parameters

fragment	- Fragment part string.

6.584.3.18 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.

6.584.3.19 void decaf::internal::net::URIType::setOpaque (bool *opaque*) [inline]

Sets if the URI is Opaque.

Parameters

opaque	true if opaque.

6.584.3.20 void decaf::internal::net::URIType::setPath (const std::string & *path*) [inline]

Sets the Path part of the URI.

Parameters

path	- Path part string.

6.584.3.21 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.

6.584.3.22 void decaf::internal::net::URIType::setQuery (const std::string & *query*) [inline]

Sets the Query part of the URI.

Parameters

query	- Query part string.

6.584.3.23 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.

6.584.3.24 void decaf::internal::net::URIType::setSchemeSpecificPart (const std::string & *schemeSpecificPart*) [inline]

Sets the Scheme Specific Part of the URI.

Parameters

schemeSpecific-Part	- scheme specific part string.

6.584.3.25 void decaf::internal::net::URIType::setServerAuthority (bool *serverAuthority*) [inline]

Sets if the URI is a Server Authority.

Parameters

serverAuthority	- true if Server Authority.

6.584.3.26 void decaf::internal::net::URIType::setSource (const std::string & *source*) [inline]

Sets the source URI string that was parsed to obtain this **URIType** (p. 2217) instance and the resulting data,.

Parameters

source	- the source URI string

6.584.3.27 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.

6.584.3.28 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. 2217) contains valid data.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIType.h**

6.585 decaf::net::URL Class Reference

Class **URL** (p. 2223) 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** ()

6.585.1 Detailed Description

Class **URL** (p. 2223) 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. 2223) can be broken into several parts. The previous example of a **URL** (p. 2223) 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. 2223) is called the path component.

A **URL** (p. 2223) 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. 2223) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 2191)): 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. 2223) 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. 2223). 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. 2223). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 2223):

`http://www.apache.org/cms/index.html`

contained within it the relative **URL** (p. 2223):

FAQ.html

it would be a shorthand for:

`http://www.apache.org/cms/FAQ.html`

The relative **URL** (p. 2223) need not specify all the components of a **URL** (p. 2223). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 2223). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 2223) class does not itself encode or decode any **URL** (p. 2223) 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. 2223), and also to decode any escaped fields, that are returned from **URL** (p. 2223). Furthermore, because **URL** (p. 2223) has no knowledge of **URL** (p. 2223) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 2223). 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. 2191) 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. 2191), and to convert between these

two classes using toURI() and **URI.toURL()** (p. 2201).

The **URLEncoder** (p. 2225) and **URLDecoder** (p. 2225) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

6.585.2 Constructor & Destructor Documentation

6.585.2.1 decaf::net::URL::URL ( )

6.585.2.2 decaf::net::URL::URL (const std::string & *url*)

6.585.2.3 virtual decaf::net::URL::~URL ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URL.h**

6.586 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.

6.586.1 Constructor & Destructor Documentation

6.586.1.1 virtual decaf::net::URLDecoder::~URLDecoder ( ) [inline, virtual]

6.586.2 Member Function Documentation

6.586.2.1 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**

6.587 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.*

6.587.1 Constructor & Destructor Documentation

6.587.1.1 virtual decaf::net::URLEncoder::~URLEncoder ( ) [inline, virtual]

6.587.2 Member Function Documentation

6.587.2.1 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-urlencode` MIME content type.

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '*', '_' are converted into their hexidecimal 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**

6.588 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. 2226) Manager.*
- virtual void **waitForSpace** (unsigned int timeout)=0

  *Waits for more space to be returned to this **Usage** (p. 2226) 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. 2226) instance is full, i.e.*

6.588.1 Constructor & Destructor Documentation

6.588.1.1 virtual activemq::util::Usage::~Usage ( ) [inline, virtual]

6.588.2 Member Function Documentation

6.588.2.1 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. 1411).

6.588.2.2 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. 1411).

6.588.2.3 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. 1411).

6.588.2.4 virtual bool activemq::util::Usage::isFull ( ) const [pure virtual]

Returns true if this **Usage** (p. 2226) instance is full, i.e.

Usage (p. 2226) >= 100%

Returns

true if **Usage** (p. 2226) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 1412).

6.588.2.5 virtual void activemq::util::Usage::waitForSpace ( ) [pure virtual]

Waits forever for more space to be returned to this **Usage** (p. 2226) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 1412).

6.588.2.6 virtual void activemq::util::Usage::waitForSpace (unsigned int *timeout*) [pure virtual]

Waits for more space to be returned to this **Usage** (p. 2226) 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. 1412).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**Usage.h**

6.589 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 ()

6.589.1 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

6.589.2 Constructor & Destructor Documentation

6.589.2.1 decaf::io::UTFDataFormatException::UTFDataFormatException ( ) throw () [inline]

Default Constructor.

6.589.2.2 decaf::io::UTFDataFormatException::UTFDataFormatException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.589.2.3 decaf::io::UTFDataFormatException::UTFDataFormatException (const UTFDataFormatException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.589.2.4 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

6.589.2.5 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.

6.589.2.6 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

6.589.2.7 virtual decaf::io::UTFDataFormatException::~UTFDataFormatException ( ) throw () [inline, virtual]

6.589.3 Member Function Documentation

6.589.3.1 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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**UTFDataFormatException.h**

6.590 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 2230)).

`#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. 2230) using the specified data.*
- virtual ∼**UUID** ()
- virtual int **compareTo** (**const UUID** &value) **const**

  *Compare the given **UUID** (p. 2230) to this one.*
- virtual bool **equals** (**const UUID** &value) **const**

  *Compares this **UUID** (p. 2230) 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. 2230).*
- virtual long long **getLeastSignificantBits** () **const**
- virtual long long **getMostSignificantBits** () **const**
- virtual long long **node** ()

  *The node value associated with this **UUID** (p. 2230).*
- virtual long long **timestamp** ()

  *The timestamp value associated with this **UUID** (p. 2230).*
- virtual int **clockSequence** ()

  *The clock sequence value associated with this **UUID** (p. 2230).*
- virtual int **variant** ()

  *The variant number associated with this **UUID** (p. 2230).*
- virtual int **version** ()

  *The version number associated with this **UUID** (p. 2230).*

Static Public Member Functions

- static **UUID randomUUID** ()

  *Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 2230).*
- static **UUID nameUUIDFromBytes** (**const** std::vector< char > &name)

  *Static factory to retrieve a type 3 (name based) **UUID** (p. 2230) 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. 2230) based on the specified byte array.*
- static **UUID fromString** (**const** std::string &name)

  *Creates a **UUID** (p. 2230) from the string standard representation as described in the **toString()** (p. 2235) method.*

6.590.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 2230)).

A **UUID** (p. 2230) 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. 2230) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 2230) is as follows: The most significant long consists of the following unsigned fields:

0xFFFFFFFF00000000 time_low 0x00000000FFFF0000 time_mid 0x000000000000F000 version 0x0000000000000-FFF time_hi

The least significant long consists of the following unsigned fields:

0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFFFFFF node

The variant field contains a value which identifies the layout of the **UUID** (p. 2230). The bit layout described above is valid only for a **UUID** (p. 2230) 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. 2230). 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.

6.590.2 Constructor & Destructor Documentation

6.590.2.1 decaf::util::UUID::UUID (long long *mostSigBits,* long long *leastSigBits*)

Constructs a new **UUID** (p. 2230) using the specified data.

mostSigBits is used for the most significant 64 bits of the **UUID** (p. 2230) and leastSigBits becomes the least significant 64 bits of the **UUID** (p. 2230).

Parameters

mostSigBits	
leastSigBits	

6.590.2.2 virtual decaf::util::UUID::~UUID ( ) [virtual]

6.590.3 Member Function Documentation

6.590.3.1 virtual int decaf::util::UUID::clockSequence ( ) [virtual]

The clock sequence value associated with this **UUID** (p. 2230).

The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 2230). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 2230).

The clockSequence value is only meaningful in a time-based **UUID** (p. 2230), which has version type 1. If this **UUID** (p. 2230) is not a time-based **UUID** (p. 2230) then this method throws UnsupportedOperationException.

Returns

the clockSequeunce associated with a V1 **UUID** (p. 2230)

Exceptions

UnsupportedOperation-Exception	if this **UUID** (p. 2230) version does not support this operation.

6.590.3.2 virtual int decaf::util::UUID::compareTo (const UUID & *value*) const [virtual]

Compare the given **UUID** (p. 2230) to this one.

Parameters

value	- the **UUID** (p. 2230) to compare to

6.590.3.3 virtual bool decaf::util::UUID::equals (const UUID & *value*) const [virtual]

Compares this **UUID** (p. 2230) to the one given, returns true if they are equal.

Parameters

value	The **UUID** (p. 2230) to compare to.

Returns

true if UUIDs are the same.

6.590.3.4 static UUID decaf::util::UUID::fromString (const std::string & *name*) [static]

Creates a **UUID** (p. 2230) from the string standard representation as described in the **toString()** (p. 2235) method.

Parameters

name	A string to be used to construct a **UUID** (p. 2230).

Returns

type 3 **UUID** (p. 2230)

Exceptions

IllegalArgumentException	if the **UUID** (p. 2230) string given is invalid.

6.590.3.5 virtual long long decaf::util::UUID::getLeastSignificantBits ( ) const [virtual]

Returns

the most significant 64 bits of this **UUID** (p. 2230)'s 128 bit value.

6.590.3.6 virtual long long decaf::util::UUID::getMostSignificantBits ( ) const [virtual]

Returns

the most significant 64 bits of this **UUID** (p. 2230)'s 128 bit value.

6.590.3.7 static UUID decaf::util::UUID::nameUUIDFromBytes (const std::vector< char > & *name*) [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 2230) based on the specified byte array.

Parameters

name	A byte array to be used to construct a **UUID** (p. 2230).

Returns

type 3 **UUID** (p. 2230)

6.590.3.8 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. 2230) based on the specified byte array.

Parameters

name	A byte array to be used to construct a **UUID** (p. 2230).
size	The size of the byte array, or number of bytes to use.

Returns

type 3 **UUID** (p. 2230)

6.590.3.9 virtual long long decaf::util::UUID::node ( ) [virtual]

The node value associated with this **UUID** (p. 2230).

The 48 bit node value is constructed from the node field of this **UUID** (p. 2230). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 2230) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 2230), which has version type 1. If this **UUID** (p. 2230) is not a time-based **UUID** (p. 2230) then this method throws UnsupportedOperationException.

Returns

the node value of this **UUID** (p. 2230)

Exceptions

UnsupportedOperation-Exception	if this **UUID** (p. 2230) version does not support this operation.

6.590.3.10 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.

6.590.3.11 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.

6.590.3.12 static UUID decaf::util::UUID::randomUUID ( ) [static]

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 2230).

The **UUID** (p. 2230) is generated using a cryptographically strong pseudo random number generator.

Returns

type 4 **UUID** (p. 2230)

6.590.3.13 virtual long long decaf::util::UUID::timestamp ( ) [virtual]

The timestamp value associated with this **UUID** (p. 2230).

The 60 bit timestamp value is constructed from the time_low, time_mid, and time_hi fields of this **UUID** (p. 2230). 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. 2230), which has version type 1. If this **UUID** (p. 2230) is not a time-based **UUID** (p. 2230) then this method throws UnsupportedOperationException.

Returns

the timestamp associated with a V1 **UUID** (p. 2230)

Exceptions

UnsupportedOperation-Exception	if this **UUID** (p. 2230) version does not support this operation.

6.590.3.14 virtual std::string decaf::util::UUID::toString ( ) const [virtual]

Returns a String object representing this **UUID** (p. 2230).

UUID (p. 2230)'s are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

Returns

formatted string for this **UUID** (p. 2230)

6.590.3.15 virtual int decaf::util::UUID::variant ( ) [virtual]

The variant number associated with this **UUID** (p. 2230).

The variant number describes the layout of the **UUID** (p. 2230). 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. 2230)

Exceptions

UnsupportedOperation-Exception	if this **UUID** (p. 2230) version does not support this operation.

6.590.3.16 virtual int decaf::util::UUID::version ( ) [virtual]

The version number associated with this **UUID** (p. 2230).

The version number describes how this **UUID** (p. 2230) was generated. The version number has the following meaning:

```
* 1 Time-based UUID
* 2 DCE security UUID
* 3 Name-based UUID
* 4 Randomly generated UUID
```

Returns

the version associated with a V1 **UUID** (p. 2230)

Exceptions

UnsupportedOperation-Exception	if this **UUID** (p. 2230) version does not support this operation.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**UUID.h**

6.591 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

  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

  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. 2236) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual bool **inReceive** () **const** =0

  Indicates if the WireFromat object is in the process of receiving a message.

- virtual **Pointer**
  < **transport::Transport** > **createNegotiator** (**const Pointer**< **transport::Transport** > &transport)=0

  If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

6.591.1 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

6.591.2 Constructor & Destructor Documentation

6.591.2.1 virtual activemq::wireformat::WireFormat::~WireFormat ( ) [inline, virtual]

6.591.3 Member Function Documentation

6.591.3.1 virtual Pointer<transport::Transport> activemq::wireformat::WireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) [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. 2251) as a **Pointer**<**Transport**> (p. 1614).

Exceptions

UnsupportedOperation-Exception	if the **WireFormat** (p. 2236) doesn't have a Negotiator.

Implemented in **activemq::wireformat::stomp::StompWireFormat** (p. 2013), and **activemq::wireformat::openwire::OpenWireFormat** (p. 1587).

6.591.3.2 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. 1588), and **activemq::wireformat::stomp::StompWireFormat** (p. 2014).

6.591.3.3 virtual bool activemq::wireformat::WireFormat::hasNegotiator ( ) const [pure virtual]

Returns true if this **WireFormat** (p. 2236) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 2236) provides a Negotiator.

Implemented in **activemq::wireformat::stomp::StompWireFormat** (p. 2014), and **activemq::wireformat::openwire::OpenWireFormat** (p. 1589).

6.591.3.4 virtual bool activemq::wireformat::WireFormat::inReceive ( ) const [pure virtual]

Indicates if the WireFromat object is in the process of receiving a message.

This is useful for monitoring inactivity and the **WireFormat** (p. 2236) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 2236) instance to determine if its busy or not and not mark the connection as inactive if so.

Returns

true if the **WireFormat** (p. 2236) object is unmarshaling a message.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1589), and **activemq::wireformat::stomp::StompWireFormat** (p. 2014).

6.591.3.5 virtual void activemq::wireformat::WireFormat::marshal (const Pointer< commands::Command > & *command,* const activemq::transport::Transport * *transport,* decaf::io::DataOutputStream * *out*) [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	if an I/O error occurs.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1590), and **activemq::wireformat::stomp::StompWireFormat** (p. 2014).

6.591.3.6 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. 1593), and **activemq::wireformat::stomp::StompWireFormat** (p. 2015).

6.591.3.7 virtual Pointer<commands::Command> activemq::wireformat::WireFormat::unmarshal (const activemq::transport::Transport * *transport,* decaf::io::DataInputStream * *in*) [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	if an I/O error occurs.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1594), and **activemq::wireformat::stomp::StompWireFormat** (p. 2015).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormat.h**

6.592 activemq::wireformat::WireFormatFactory Class Reference

The **WireFormatFactory** (p. 2239) is the interface that all **WireFormatFactory** (p. 2239) 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

  *Creates a new **WireFormat** (p. 2236) Object passing it a set of properties from which it can obtain any optional settings.*

6.592.1 Detailed Description

The **WireFormatFactory** (p. 2239) is the interface that all **WireFormatFactory** (p. 2239) classes must extend.

The Factory creates a **WireFormat** (p. 2236) Object based on the properties that are set in the passed `Properties` object.

6.592.2 Constructor & Destructor Documentation

6.592.2.1 virtual activemq::wireformat::WireFormatFactory::~WireFormatFactory ( ) [inline, virtual]

6.592.3 Member Function Documentation

6.592.3.1 virtual Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) [pure virtual]

Creates a new **WireFormat** (p. 2236) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties	The Properties for this **WireFormat** (p. 2236).

Returns

Pointer to a new instance of a **WireFormat** (p. 2236) object.

Exceptions

IllegalStateException	if the factory has not been initialized.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 1595), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 2016).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatFactory.h**

6.593 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 **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **DataStructure** * **cloneDataStructure** () **const**

  Clone this obbject 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. *877) such as its type and value of its elements.*

- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure*** (p. *877) passed in to this one, and returns if they are equivalent.*

- virtual bool **isMarshalAware** () **const**

  Determine if the class implementing this interface is really wanting to be told about marshaling.

- 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 **getMaxInactivityDurationInitalDelay** () **const**

  Returns the currently configured Max Inactivity Intial Delay duration.

- void **setMaxInactivityDurationInitalDelay** (long long maxInactivityDurationInitalDelay)

  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. 671).*
- virtual **util::PrimitiveMap** & **getProperties** ()

  *Gets the Properties for this **Command** (p. 671).*
- virtual void **setProperties** (**const util::PrimitiveMap** &map)

  *Sets the Properties for this **Command** (p. 671).*
- bool **isValid** () **const**

  *Determines if we think this is a Valid **WireFormatInfo** (p. 2240) command.*
- virtual bool **isWireFormatInfo** () **const**
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)
- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)

Static Public Attributes

- static **const** unsigned char **ID_WIREFORMATINFO** = 1

6.593.1 Constructor & Destructor Documentation

6.593.1.1 activemq::commands::WireFormatInfo::WireFormatInfo ( )

6.593.1.2 virtual activemq::commands::WireFormatInfo::~WireFormatInfo ( ) [virtual]

6.593.2 Member Function Documentation

6.593.2.1 virtual void activemq::commands::WireFormatInfo::afterUnmarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) [virtual]

Reimplemented from **activemq::commands::BaseDataStructure** (p. 410).

6.593.2.2 virtual void activemq::commands::WireFormatInfo::beforeMarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) [virtual]

Reimplemented from **activemq::commands::BaseDataStructure** (p. 410).

6.593.2.3 virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure ( ) const [virtual]

Clone this obbject 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. 878).

6.593.2.4 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. 382).

6.593.2.5 virtual bool activemq::commands::WireFormatInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 382).

6.593.2.6 int activemq::commands::WireFormatInfo::getCacheSize ( ) const

Gets the Cache Size setting.

Returns

currently set cache size.

6.593.2.7 virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 880).

6.593.2.8 const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic ( ) const [inline]

Get the Magic field.

Returns

const reference to a std::vector<char>

6.593.2.9 const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshalledProperties ( ) const [inline]

Get the marshalledProperties field.

Returns

const reference to a std::vector<char>

6.593.2.10 long long activemq::commands::WireFormatInfo::getMaxInactivityDuration ( ) const

Returns the currently configured Max Inactivity duration.

Returns

the set inactivity duration value.

6.593.2.11 long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitalDelay ( ) const

Returns the currently configured Max Inactivity Intial Delay duration.

Returns

the set inactivity duration initial delay value.

6.593.2.12 virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ( ) const [inline, virtual]

Gets the Properties for this **Command** (p. 671).

Returns

the Properties object for this **Command** (p. 671).

6.593.2.13 virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ( ) [inline, virtual]

Gets the Properties for this **Command** (p. 671).

Returns

the Properties object for this **Command** (p. 671).

6.593.2.14 int activemq::commands::WireFormatInfo::getVersion ( ) const [inline]

Get the current Wireformat Version.

Returns

int that identifies the version

6.593.2.15 bool activemq::commands::WireFormatInfo::isCacheEnabled ( ) const

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.593.2.16 virtual bool activemq::commands::WireFormatInfo::isMarshalAware ( ) const [inline, 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.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 410).

6.593.2.17 bool activemq::commands::WireFormatInfo::isSizePrefixDisabled ( ) const

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.593.2.18 bool activemq::commands::WireFormatInfo::isStackTraceEnabled ( ) const

Checks if the stackTraceEnabled flag is on.

Returns

true if the flag is on.

6.593.2.19 bool activemq::commands::WireFormatInfo::isTcpNoDelayEnabled ( ) const

Checks if the tcpNoDelayEnabled flag is on.

Returns

true if the flag is on.

6.593.2.20 bool activemq::commands::WireFormatInfo::isTightEncodingEnabled ( ) const

Checks if the tightEncodingEnabled flag is on.

Returns

true if the flag is on.

6.593.2.21 bool activemq::commands::WireFormatInfo::isValid ( ) const

Determines if we think this is a Valid **WireFormatInfo** (p. 2240) command.

Returns

true if its valid.

6.593.2.22 virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo ( ) const [inline, virtual]

Returns

answers true to the isWireFormatInfo query

Reimplemented from **activemq::commands::BaseCommand** (p. 385).

6.593.2.23 void activemq::commands::WireFormatInfo::setCacheEnabled (bool *cacheEnabled*)

Sets if the cacheEnabled flag is on.

Parameters

cacheEnabled	- true to turn flag is on

6.593.2.24 void activemq::commands::WireFormatInfo::setCacheSize (int *value*)

Sets the Cache Size setting.

Parameters

value	- value to set to the cache size.

6.593.2.25 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>

6.593.2.26 void activemq::commands::WireFormatInfo::setMarshalledProperties (const std::vector< unsigned char > & *marshalledProperties*) [inline]

Sets the value of the marshalledProperties field.

Parameters

marshalled-Properties	The Byte Array vector that contains the marshaled form of the **Message** (p. 1412) properties, this is the data sent over the wire.

6.593.2.27 void activemq::commands::WireFormatInfo::setMaxInactivityDuration (long long *maxInactivityDuration*)

Sets the Max inactivity duration value.

Parameters

maxInactivity-Duration	- max time a client can be inactive.

6.593.2.28 void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitalDelay (long long *maxInactivityDurationInitalDelay*)

Sets the Max inactivity initial delay duration value.

Parameters

maxInactivity-DurationInital-Delay	- time before the inactivity delay is checked.

6.593.2.29 virtual void activemq::commands::WireFormatInfo::setProperties (const util::PrimitiveMap & *map*) [inline, virtual]

Sets the Properties for this **Command** (p. 671).

Parameters

map	- PrimtiveMap to copy

6.593.2.30 void activemq::commands::WireFormatInfo::setSizePrefixDisabled (bool *sizePrefixDisabled*)

Sets if the sizePrefixDisabled flag is on.

Parameters

sizePrefix-Disabled	- true to turn flag is on

6.593.2.31 void activemq::commands::WireFormatInfo::setStackTraceEnabled (bool *stackTraceEnabled*)

Sets if the stackTraceEnabled flag is on.

Parameters

stackTrace-Enabled	- ture to turn flag is on

6.593.2.32 void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*)

Sets if the tcpNoDelayEnabled flag is on.

Parameters

tcpNoDelay-Enabled	- ture to turn flag is on

6.593.2.33 void activemq::commands::WireFormatInfo::setTightEncodingEnabled (bool *tightEncodingEnabled*)

Sets if the tightEncodingEnabled flag is on.

Parameters

tightEncoding-Enabled	- true to turn flag is on

6.593.2.34 void activemq::commands::WireFormatInfo::setVersion (int *version*) [inline]

Set the current Wireformat Version.

Parameters

version	- int that identifies the version

6.593.2.35 virtual std::string activemq::commands::WireFormatInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 386).

6.593.2.36 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. 1781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 674).

6.593.3 Field Documentation

6.593.3.1 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**

6.594 activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2248).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/WireFormatInfoMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual ∼**WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure** ∗ **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn, **utils::BooleanStream** ∗bs)

  Tight Un-marhsal to the given stream.
- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.
- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.594.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2248).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.594.2 Constructor & Destructor Documentation

6.594.2.1 activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]

6.594.2.2 virtual activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::∼WireFormatInfoMarshaller ( ) [inline, virtual]

6.594.3 Member Function Documentation

6.594.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.594.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.594.3.3 virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 871).

6.594.3.4 virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 872).

6.594.3.5 virtual int activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 874).

6.594.3.6 virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 875).

6.594.3.7 virtual void activemq::wireformat::openwire::marshal::generated::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 876).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**WireFormatInfoMarshaller.h**

6.595 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 2251) which allows a **WireFormat** (p. 2236) to.

`#include <src/main/activemq/wireformat/WireFormatNegotiator.h>`

Inheritance diagram for activemq::wireformat::WireFormatNegotiator:

Public Member Functions

- **WireFormatNegotiator** (**const Pointer**< **transport::Transport** > &**next**)
  *Creates a new instance of a **WireFormat** (p. 2236) Negotiator wrapping the Transport passed.*
- virtual ~**WireFormatNegotiator** ()

6.595.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 2251) which allows a **WireFormat** (p. 2236) to.

6.595.2 Constructor & Destructor Documentation

6.595.2.1 activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator (const Pointer< transport::Transport > & *next*) [inline]

Creates a new instance of a **WireFormat** (p. 2236) Negotiator wrapping the Transport passed.

Parameters

next	The next Transport in the chain

6.595.2.2 virtual activemq::wireformat::WireFormatNegotiator::~WireFormatNegotiator ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatNegotiator.h**

6.596 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 2236) 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**

  *Gets a Registered **WireFormatFactory** (p. 2239) 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)

  *Registers a new **WireFormatFactory** (p. 2239) with this Registry.*
- void **unregisterFactory** (**const** std::string &name)

  Unregisters the Factory with the given name and deletes that instance of the Factory.
- void **unregisterAllFactories** ()

  Removes all Factories and deletes the instances of the Factory objects.
- std::vector< std::string > **getWireFormatNames** () **const**

  *Retrieves a list of the names of all the Registered **WireFormat** (p. 2236)'s in this Registry.*

Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()

  *Gets the single instance of the **WireFormatRegistry** (p. 2252).*

6.596.1 Detailed Description

Registry of all **WireFormat** (p. 2236) Factories that are available to the client at runtime.

New **WireFormat** (p. 2236)'s must have a factory registered here before a connection attempt is made.

Since

3.0

6.596.2 Constructor & Destructor Documentation

6.596.2.1 virtual activemq::wireformat::WireFormatRegistry::~WireFormatRegistry ( ) [virtual]

6.596.3 Member Function Documentation

6.596.3.1 WireFormatFactory* activemq::wireformat::WireFormatRegistry::findFactory (const std::string & *name*) const

Gets a Registered **WireFormatFactory** (p. 2239) 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.

6.596.3.2 static WireFormatRegistry& activemq::wireformat::WireFormatRegistry::getInstance ( ) [static]

Gets the single instance of the **WireFormatRegistry** (p. 2252).

Returns

reference to the single instance of this Registry

6.596.3.3 std::vector<std::string> activemq::wireformat::WireFormatRegistry::getWireFormatNames ( ) const

Retrieves a list of the names of all the Registered **WireFormat** (p. 2236)'s in this Registry.

Returns

stl vector of strings with all the **WireFormat** (p. 2236) names registered.

6.596.3.4 void activemq::wireformat::WireFormatRegistry::registerFactory (const std::string & *name,* WireFormatFactory * *factory*)

Registers a new **WireFormatFactory** (p. 2239) 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.

6.596.3.5 void activemq::wireformat::WireFormatRegistry::unregisterAllFactories ( )

Removes all Factories and deletes the instances of the Factory objects.

6.596.3.6 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**

6.597 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.

6.597.1 Detailed Description

Runnable class used by the {.

See also

InactivityMonitor (p. 1104)} to make periodic writes to the underlying **transport** (p. 56) 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

6.597.2 Constructor & Destructor Documentation

6.597.2.1 activemq::transport::inactivity::WriteChecker::WriteChecker (InactivityMonitor ∗ *parent*)

6.597.2.2 virtual activemq::transport::inactivity::WriteChecker::∼WriteChecker ( ) [virtual]

6.597.3 Member Function Documentation

6.597.3.1 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. 1793).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**WriteChecker.h**

6.598 decaf::io::Writer Class Reference

`#include <src/main/decaf/io/Writer.h>`

Inheritance diagram for decaf::io::Writer:

Public Member Functions

- **Writer** ()
- virtual ∼**Writer** ()
- virtual void **write** (char v)

  Writes an single byte char value.
- virtual void **write** (**const** std::vector< char > &buffer)

  Writes an array of Chars.
- virtual void **write** (**const** char *buffer, int size)

  Writes a byte array to the output stream.
- virtual void **write** (**const** char *buffer, int size, int offset, int length)

  Writes a byte array to the output stream.
- virtual void **write** (**const** std::string &str)

  Writes a string.
- virtual void **write** (**const** std::string &str, int offset, int length)

  Writes a string.
- virtual **decaf::lang::Appendable** & **append** (char value)

  Appends the specified character to this Appendable.
- virtual **decaf::lang::Appendable** & **append** (**const decaf::lang::CharSequence** *csq)

  Appends the specified character sequence to this Appendable.
- virtual **decaf::lang::Appendable** & **append** (**const decaf::lang::CharSequence** *csq, int start, int end)

  Appends a subsequence of the specified character sequence to this Appendable.

Protected Member Functions

- virtual void **doWriteArrayBounded** (**const** char *buffer, int size, int offset, int length)=0

  Override this method to customize the functionality of the method write(char buffer, int size, int offset, int length).*
- virtual void **doWriteChar** (char v)
- virtual void **doWriteVector** (**const** std::vector< char > &buffer)
- virtual void **doWriteArray** (**const** char *buffer, int size)
- virtual void **doWriteString** (**const** std::string &str)
- virtual void **doWriteStringBounded** (**const** std::string &str, int offset, int length)
- virtual **decaf::lang::Appendable** & **doAppendChar** (char value)
- virtual **decaf::lang::Appendable** & **doAppendCharSequence** (**const decaf::lang::CharSequence** *csq)
- virtual **decaf::lang::Appendable** & **doAppendCharSequenceStartEnd** (**const decaf::lang::CharSequence** *csq, int start, int end)

6.598.1 Constructor & Destructor Documentation

6.598.1.1 decaf::io::Writer::Writer ( )

6.598.1.2 virtual decaf::io::Writer::∼Writer ( ) [virtual]

6.598.2 Member Function Documentation

6.598.2.1 virtual decaf::lang::Appendable& decaf::io::Writer::append (char *value*) [virtual]

Appends the specified character to this Appendable.

Parameters

value	The character to append.

Returns

a Reference to this Appendable

Exceptions

Exception	if an error occurs.

Implements **decaf::lang::Appendable** (p. 343).

6.598.2.2 virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence * *csq*) [virtual]

Appends the specified character sequence to this Appendable.

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.

Exceptions

Exception	if an error occurs.

Implements **decaf::lang::Appendable** (p. 343).

6.598.2.3 virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence * *csq,* int *start,* int *end*) [virtual]

Appends a subsequence of the specified character sequence to this Appendable.

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

Exceptions

Exception	if an error occurs.
IndexOutOfBounds-Exception	start is greater than end, or end is greater than csq.length()

Implements **decaf::lang::Appendable** (p. 344).

6.598.2.4 virtual decaf::lang::Appendable& decaf::io::Writer::doAppendChar (char *value*) [protected, virtual]

6.598.2.5 virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequence (const decaf::lang::CharSequence * *csq*) [protected, virtual]

6.598.2.6 virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequenceStartEnd (const decaf::lang::CharSequence * *csq,* int *start,* int *end*) [protected, virtual]

6.598.2.7 virtual void decaf::io::Writer::doWriteArray (const char * *buffer,* int *size*) [protected, virtual]

6.598.2.8 virtual void decaf::io::Writer::doWriteArrayBounded (const char * *buffer,* int *size,* int *offset,* int *length*) [protected, pure virtual]

Override this method to customize the functionality of the method write(char* buffer, int size, int offset, int length).

All subclasses must override this method to provide the basic **Writer** (p. 2255) functionality.

Implemented in **decaf::io::OutputStreamWriter** (p. 1608).

6.598.2.9 virtual void decaf::io::Writer::doWriteChar (char *v*) [protected, virtual]

6.598.2.10 virtual void decaf::io::Writer::doWriteString (const std::string & *str*) [protected, virtual]

6.598.2.11 virtual void decaf::io::Writer::doWriteStringBounded (const std::string & *str,* int *offset,* int *length*) [protected, virtual]

6.598.2.12 virtual void decaf::io::Writer::doWriteVector (const std::vector< char > & *buffer*) [protected, virtual]

6.598.2.13 virtual void decaf::io::Writer::write (char *v*) [virtual]

Writes an single byte char value.

Parameters

v	The value to be written.

Exceptions

IOException *(p. 1198)*	thrown if an error occurs.

6.598.2.14 virtual void decaf::io::Writer::write (const std::vector< char > & *buffer*) [virtual]

Writes an array of Chars.

Parameters

buffer	The array to be written.

Exceptions

IOException *(p. 1198)*	thrown if an error occurs.

6.598.2.15 virtual void decaf::io::Writer::write (const char * *buffer,* int *size*) [virtual]

Writes a byte array to the output stream.

Parameters

buffer	The byte array to write (cannot be NULL).
size	The size in bytes of the buffer passed.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.
NullPointerException	if buffer is NULL.

6.598.2.16 virtual void decaf::io::Writer::write (const char * *buffer,* int *size,* int *offset,* int *length*) [virtual]

Writes a byte array to the output stream.

Parameters

buffer	The byte array to write (cannot be NULL).
size	The size in bytes of the buffer passed.
offset	The position in the array to start writing from.
length	The number of bytes in the array to write.

Exceptions

IOException *(p. 1198)*	if an I/O error occurs.
NullPointerException	if buffer is NULL.
IndexOutOfBounds-Exception	if offset + length $>$ size of the buffer.

6.598.2.17 virtual void decaf::io::Writer::write (const std::string & *str*) [virtual]

Writes a string.

Parameters

str	The string to be written.

Exceptions

IOException *(p. 1198)*	thrown if an error occurs.

6.598.2.18 virtual void decaf::io::Writer::write (const std::string & *str,* int *offset,* int *length*) [virtual]

Writes a string.

Parameters

str	The string to be written.
offset	The position in the array to start writing from.
length	The number of bytes in the array to write.

Exceptions

IOException *(p. 1198)*	thrown if an error occurs.
IndexOutOfBounds-Exception	if offset+length is greater than the string length.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Writer.h**

6.599 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

6.599.1 Constructor & Destructor Documentation

6.599.1.1 virtual decaf::security::auth::x500::X500Principal::~X500Principal ( ) [inline, virtual]

6.599.2 Member Function Documentation

6.599.2.1 virtual void decaf::security::auth::x500::X500Principal::getEncoded (std::vector< unsigned char > & *output*) const [pure virtual]

6.599.2.2 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. 1674).

6.599.2.3 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**

6.600 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
- virtual void **checkValidity** (**const decaf::util::Date** &date) **const** =0
- 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
- virtual int **getVersion** () **const** =0

6.600.1 Detailed Description

Base interface for all identity certificates.

6.600.2 Constructor & Destructor Documentation

6.600.2.1 virtual decaf::security::cert::X509Certificate::~X509Certificate ( ) [inline, virtual]

6.600.3 Member Function Documentation

6.600.3.1 virtual void decaf::security::cert::X509Certificate::checkValidity ( ) const [pure virtual]

6.600.3.2 virtual void decaf::security::cert::X509Certificate::checkValidity (const decaf::util::Date & *date*) const [pure virtual]

6.600.3.3 virtual int decaf::security::cert::X509Certificate::getBasicConstraints ( ) const [pure virtual]

6.600.3.4 virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID (std::vector< bool > & *output*) const [pure virtual]

6.600.3.5 virtual const X500Principal* decaf::security::cert::X509Certificate::getIssuerX500Principal ( ) const [pure virtual]

6.600.3.6 virtual void **decaf::security::cert::X509Certificate::getKeyUsage** (std::vector< unsigned char > & *output*) const [pure virtual]

6.600.3.7 virtual Date **decaf::security::cert::X509Certificate::getNotAfter** ( ) const [pure virtual]

6.600.3.8 virtual Date **decaf::security::cert::X509Certificate::getNotBefore** ( ) const [pure virtual]

6.600.3.9 virtual std::string **decaf::security::cert::X509Certificate::getSigAlgName** ( ) const [pure virtual]

6.600.3.10 virtual std::string **decaf::security::cert::X509Certificate::getSigAlgOID** ( ) const [pure virtual]

6.600.3.11 virtual void **decaf::security::cert::X509Certificate::getSigAlgParams** (std::vector< unsigned char > & *output*) const [pure virtual]

6.600.3.12 virtual void **decaf::security::cert::X509Certificate::getSignature** (std::vector< unsigned char > & *output*) const [pure virtual]

6.600.3.13 virtual void **decaf::security::cert::X509Certificate::getSubjectUniqueID** (std::vector< bool > & *output*) const [pure virtual]

6.600.3.14 virtual const X500Principal* **decaf::security::cert::X509Certificate::getSubjectX500Principal** ( ) const [pure virtual]

6.600.3.15 virtual void **decaf::security::cert::X509Certificate::getTBSCertificate** (std::vector< unsigned char > & *output*) const [pure virtual]

6.600.3.16 virtual int **decaf::security::cert::X509Certificate::getVersion** ( ) const [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**X509Certificate.h**

6.601 cms::XAConnection Class Reference

The **XAConnection** (p. 2262) interface defines an extended **Connection** (p. 725) type that is used to create **XASession** (p. 2275) objects.

`#include <src/main/cms/XAConnection.h>`

Inheritance diagram for cms::XAConnection:

Public Member Functions

- virtual ~**XAConnection** () throw ()
- virtual **XASession** * **createXASession** ()=0

  *Creates an **XASession** (p. 2275) object.*

6.601.1 Detailed Description

The **XAConnection** (p. 2262) interface defines an extended **Connection** (p. 725) type that is used to create **XASession** (p. 2275) objects.

This is an optional interface and CMS providers are allowed to omit an implementation and instead throw an exception from an **XAConnectionFactory** (p. 2263) stub to indicate that XA is not supported.

Since

2.3

6.601.2 Constructor & Destructor Documentation

6.601.2.1 virtual cms::XAConnection::~XAConnection ( ) throw () [virtual]

6.601.3 Member Function Documentation

6.601.3.1 virtual XASession* cms::XAConnection::createXASession ( ) [pure virtual]

Creates an **XASession** (p. 2275) object.

Returns

a newly created **XASession** (p. 2275) instance, caller owns the pointer.

Exceptions

CMSException *(p. 640)*	If the **XAConnection** (p. 2262) object fails to create the **XASession** (p. 2275) instance due to an internal error.

Implemented in **activemq::core::ActiveMQXAConnection** (p. 336).

The documentation for this class was generated from the following file:

- src/main/cms/**XAConnection.h**

6.602 cms::XAConnectionFactory Class Reference

The **XAConnectionFactory** (p. 2263) interface is specialized interface that defines an **ConnectionFactory** (p. 741) that creates **Connection** (p. 725) instance that will participate in XA Transactions.

`#include <src/main/cms/XAConnectionFactory.h>`

Inheritance diagram for cms::XAConnectionFactory:

Public Member Functions

- virtual ~**XAConnectionFactory** ()
- virtual **XAConnection** * **createXAConnection** ()=0

  *Creates an **XAConnection** (p. 2262) with the default user name and password.*
- virtual **XAConnection** * **createXAConnection** (**const** std::string &userName, **const** std::string &password)=0

  Creates an XA connection with the specified user name and password.

Static Public Member Functions

- static **XAConnectionFactory** * **createCMSXAConnectionFactory** (**const** std::string &brokerURI)

Static method that is used to create a provider specific XA ***Connection*** *(p. 725) factory.*

6.602.1 Detailed Description

The **XAConnectionFactory** (p. 2263) interface is specialized interface that defines an **ConnectionFactory** (p. 741) that creates **Connection** (p. 725) instance that will participate in XA Transactions.

Some application provide support for grouping XA capable resource use into a distributed transaction (optional). To include CMS API transactions in a XA transaction, an application requires a XA aware library. A CMS provider exposes its XA support using an **XAConnectionFactory** (p. 2263) object, which an application uses to create **XAConnection** (p. 2262) objects.

The **XAConnectionFactory** (p. 2263) interface is optional. CMS providers are not required to support this interface. This interface is for use by CMS providers to support transactional environments. Client programs are strongly encouraged to use the transactional support available in their environment, rather than use these XA interfaces directly.

Since

2.3

6.602.2 Constructor & Destructor Documentation

6.602.2.1 virtual cms::XAConnectionFactory::~XAConnectionFactory ( ) [virtual]

6.602.3 Member Function Documentation

6.602.3.1 static XAConnectionFactory* cms::XAConnectionFactory::createCMSXAConnectionFactory (const std::string & *brokerURI*) [static]

Static method that is used to create a provider specific XA **Connection** (p. 725) factory.

The provider implements this method in their library and returns an instance of a **XAConnectionFactory** (p. 2263) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

The XA interfaces are optional in CMS however if a provider chooses to omit them it should still override this method and throw an **UnsupportedOperationException** (p. 2190) to indicate that it doesn't provide this functionality.

Parameters

brokerURI	The remote address to use to connect to the Provider.

Returns

A pointer to a provider specific implementation of the **XAConnectionFactory** (p. 2263) interface, the caller is responsible for deleting this resource.

Exceptions

CMSException *(p. 640)*	if an internal error occurs while creating the **XAConnectionFactory** (p. 2263).
UnsupportedOperationException *(p. 2190)*	if the provider does not support the XA API.

6.602.3.2 virtual XAConnection* cms::XAConnectionFactory::createXAConnection ( ) [pure virtual]

Creates an **XAConnection** (p. 2262) with the default user name and password.

The connection is created in stopped mode just as the standard **Connection** (p. 725) object is created from the **ConnectionFactory** (p. 741). No messages will be delivered until the **Connection.start** (p. 1964) method is explicitly called.

Returns

a new **XAConnectionFactory** (p. 2263) instance, the caller owns the returned pointer.

Exceptions

CMSException *(p. 640)*	if an internal error occurs while creating the **Connection** (p. 725).
CMSSecurityException *(p. 648)*	if the client authentication fails because the user name or password are invalid.

Implemented in **activemq::core::ActiveMQXAConnectionFactory** (p. 337).

6.602.3.3 virtual XAConnection* cms::XAConnectionFactory::createXAConnection (const std::string & *userName,* const std::string & *password*) [pure virtual]

Creates an XA connection with the specified user name and password.

The connection is created in stopped mode just as the standard **ConnectionFactory** (p. 741) creates a new **Connection** (p. 725). No messages will be delivered until the **Connection.start** (p. 1964) method is explicitly called.

Returns

a new **XAConnectionFactory** (p. 2263) instance, the caller owns the returned pointer.

Exceptions

CMSException *(p. 640)*	if an internal error occurs while creating the **Connection** (p. 725).
CMSSecurityException *(p. 648)*	if the client authentication fails because the user name or password are invalid.

Implemented in **activemq::core::ActiveMQXAConnectionFactory** (p. 338).

The documentation for this class was generated from the following file:

- src/main/cms/**XAConnectionFactory.h**

6.603 cms::XAException Class Reference

The **XAException** (p. 2265) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.

`#include <src/main/cms/XAException.h>`

Inheritance diagram for cms::XAException:

Public Member Functions

- **XAException** ()
- **XAException** (int errorCode)
- **XAException** (**const XAException** &ex)

- **XAException** (**const** std::string &message)
- **XAException** (**const** std::string &message, **const** std::exception ∗cause)
- **XAException** (**const** std::string &message, **const** std::exception ∗cause, **const** std::vector< std::pair< std-::string, int > > &stackTrace)
- virtual ∼**XAException** () throw ()
- void **setErrorCode** (int errorCode)

  *Sets the error code for this **XAException** (p. 2265).*
- int **getErrorCode** () **const**

  *Gets the error code that was assigned to this **XAException** (p. 2265).*

Static Public Attributes

- static **const** int **XA_RBBASE**

  Code which contains the inclusive lower bound of the rollback error codes.
- static **const** int **XA_RBROLLBACK**

  Code which means that the rollback occurred for an unspecified reason.
- static **const** int **XA_RBCOMMFAIL**

  Code which means that rollback was caused by a communication failure.
- static **const** int **XA_RBDEADLOCK**

  Code which means that a failure occurred because a deadlock was detected.
- static **const** int **XA_RBINTEGRITY**

  Code which means that a condition was detected than implies a violation of the integrity of the resource.
- static **const** int **XA_RBOTHER**

  Code which means that the Resource Manager rolled back the transaction branch for a reason not separately listed.
- static **const** int **XA_RBPROTO**

  Code which means that a protocol error occurred in the Resource Manager.
- static **const** int **XA_RBTIMEOUT**

  Code which means that a transaction branch took too long.
- static **const** int **XA_RBTRANSIENT**

  Code which means that the caller may retry the transaction branch.
- static **const** int **XA_RBEND**

  Code which contains the inclusive upper bound of the rollback error codes.
- static **const** int **XA_NOMIGRATE**

  Code which means that resumption must occur where the suspension occurred.
- static **const** int **XA_HEURHAZ**

  Code which means that the transaction branch may have been heuristically completed.
- static **const** int **XA_HEURCOM**

  Code which means that the transaction branch has been heuristically committed.
- static **const** int **XA_HEURRB**

  Code which means that the transaction branch has been heuristically rolled back.
- static **const** int **XA_HEURMIX**

  Code which means that the transaction branch has been heuristically committed and rolled back.
- static **const** int **XA_RETRY**

  Code which means that the method returned with no effect and can be reissued.
- static **const** int **XA_RDONLY**

  Code which means that the transaction branch was read only and has been committed.
- static **const** int **XAER_ASYNC**

  Code which means that there is already an asynchronous operation outstanding.
- static **const** int **XAER_RMERR**

  Code which means that a Resource Manager error has occurred for the transaction branch.
- static **const** int **XAER_NOTA**

Code which means that the XID is not valid.

- static **const** int **XAER_INVAL**

  Code which means that invalid arguments were supplied.

- static **const** int **XAER_PROTO**

  Code which means that the method was invoked in an improper context.

- static **const** int **XAER_RMFAIL**

  Code which means that the Resource Manager is unavailable.

- static **const** int **XAER_DUPID**

  Code which means that the XID already exists.

- static **const** int **XAER_OUTSIDE**

  Work is being done by the Resource Manager outside the boundaries of a global transaction.

6.603.1 Detailed Description

The **XAException** (p. 2265) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.

Since

2.3

6.603.2 Constructor & Destructor Documentation

6.603.2.1 cms::XAException::XAException ( )

6.603.2.2 cms::XAException::XAException (int *errorCode*)

6.603.2.3 cms::XAException::XAException (const XAException & *ex*)

6.603.2.4 cms::XAException::XAException (const std::string & *message*)

6.603.2.5 cms::XAException::XAException (const std::string & *message,* const std::exception * *cause*)

6.603.2.6 cms::XAException::XAException (const std::string & *message,* const std::exception * *cause,* const std::vector< std::pair< std::string, int > > & *stackTrace*)

6.603.2.7 virtual cms::XAException::~XAException ( ) throw () [virtual]

6.603.3 Member Function Documentation

6.603.3.1 int cms::XAException::getErrorCode ( ) const [inline]

Gets the error code that was assigned to this **XAException** (p. 2265).

Returns

the assigned error code.

6.603.3.2 void cms::XAException::setErrorCode (int *errorCode*) [inline]

Sets the error code for this **XAException** (p. 2265).

Parameters

errorCode	The error code to assign to this **XAException** (p. 2265).

6.603.4 Field Documentation

6.603.4.1 const int cms::XAException::XA_HEURCOM [static]

Code which means that the transaction branch has been heuristically committed.

6.603.4.2 const int cms::XAException::XA_HEURHAZ [static]

Code which means that the transaction branch may have been heuristically completed.

6.603.4.3 const int cms::XAException::XA_HEURMIX [static]

Code which means that the transaction branch has been heuristically committed and rolled back.

6.603.4.4 const int cms::XAException::XA_HEURRB [static]

Code which means that the transaction branch has been heuristically rolled back.

6.603.4.5 const int cms::XAException::XA_NOMIGRATE [static]

Code which means that resumption must occur where the suspension occurred.

6.603.4.6 const int cms::XAException::XA_RBBASE [static]

Code which contains the inclusive lower bound of the rollback error codes.

6.603.4.7 const int cms::XAException::XA_RBCOMMFAIL [static]

Code which means that rollback was caused by a communication failure.

6.603.4.8 const int cms::XAException::XA_RBDEADLOCK [static]

Code which means that a failure occurred because a deadlock was detected.

6.603.4.9 const int cms::XAException::XA_RBEND [static]

Code which contains the inclusive upper bound of the rollback error codes.

6.603.4.10 const int cms::XAException::XA_RBINTEGRITY [static]

Code which means that a condition was detected than implies a violation of the integrity of the resource.

6.603.4.11 const int cms::XAException::XA_RBOTHER [static]

Code which means that the Resource Manager rolled back the transaction branch for a reason not separately listed.

6.603.4.12 const int cms::XAException::XA_RBPROTO [static]

Code which means that a protocol error occurred in the Resource Manager.

6.603.4.13 const int cms::XAException::XA_RBROLLBACK [static]

Code which means that the rollback occurred for an unspecified reason.

6.603.4.14 const int cms::XAException::XA_RBTIMEOUT [static]

Code which means that a transaction branch took too long.

6.603.4.15 const int cms::XAException::XA_RBTRANSIENT [static]

Code which means that the caller may retry the transaction branch.

6.603.4.16 const int cms::XAException::XA_RDONLY [static]

Code which means that the transaction branch was read only and has been committed.

6.603.4.17 const int cms::XAException::XA_RETRY [static]

Code which means that the method returned with no effect and can be reissued.

6.603.4.18 const int cms::XAException::XAER_ASYNC [static]

Code which means that there is already an asynchronous operation outstanding.

6.603.4.19 const int cms::XAException::XAER_DUPID [static]

Code which means that the XID already exists.

6.603.4.20 const int cms::XAException::XAER_INVAL [static]

Code which means that invalid arguments were supplied.

6.603.4.21 const int cms::XAException::XAER_NOTA [static]

Code which means that the XID is not valid.

6.603.4.22 const int cms::XAException::XAER_OUTSIDE [static]

Work is being done by the Resource Manager outside the boundaries of a global transaction.

6.603.4.23 const int cms::XAException::XAER_PROTO [static]

Code which means that the method was invoked in an improper context.

6.603.4.24 const int cms::XAException::XAER_RMERR [static]

Code which means that a Resource Manager error has occurred for the transaction branch.

6.603.4.25 const int cms::XAException::XAER_RMFAIL [static]

Code which means that the Resource Manager is unavailable.

The documentation for this class was generated from the following file:

- src/main/cms/**XAException.h**

6.604 cms::XAResource Class Reference

The **XAResource** (p. 2269) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).

`#include <src/main/cms/XAResource.h>`

Inheritance diagram for cms::XAResource:

Public Member Functions

- virtual ~**XAResource** ()
- virtual void **commit** (**const Xid** *xid, bool onePhase)=0

  Commits a global transaction.
- virtual void **end** (**const Xid** *xid, int flags)=0

  Ends the work done for a transaction branch.
- virtual void **forget** (**const Xid** *xid)=0

  Informs the Resource Manager that it can forget about a specified transaction branch.
- virtual int **getTransactionTimeout** () **const** =0

  *Gets the transaction timeout value for this **XAResource** (p. 2269).*
- virtual bool **isSameRM** (**const XAResource** *theXAResource)=0

  *Returns true if the ResourceManager for this **XAResource** (p. 2269) is the same as the Resource Manager for a supplied **XAResource** (p. 2269).*
- virtual int **prepare** (**const Xid** *xid)=0

  Requests the Resource manager to prepare to commit a specified transaction.
- virtual int **recover** (int flag, **Xid** **recovered)=0

  Get a list of prepared transaction branches.
- virtual void **rollback** (**const Xid** *xid)=0

  Requests the Resource Manager to rollback a specified transaction branch.
- virtual bool **setTransactionTimeout** (int seconds)=0

  *Sets the transaction timeout value for this **XAResource** (p. 2269).*
- virtual void **start** (**const Xid** *xid, int flags)=0

  Starts work for a specified transaction branch.

Static Public Attributes

- static **const** int **TMENDRSCAN**

  Flag to end a recovery scan.
- static **const** int **TMFAIL**

  Flag to indicate that the caller is dissociation from a transaction branch and that it should be marked rollback only.
- static **const** int **TMJOIN**

  Flag to indicate that the caller is joining sn existing transaction branch.

- static **const** int **TMNOFLAGS**

  Flag that indicates that no flags options are selected.
- static **const** int **TMONEPHASE**

  Flag that indicates the caller is using one-phase commit optimization.
- static **const** int **TMRESUME**

  Flag that indicates the caller is resuming association with a suspended transaction branch.
- static **const** int **TMSTARTRSCAN**

  Flag that indicates the start of a recovery scan.
- static **const** int **TMSUCCESS**

  Flag that indicates the caller is dissociating from a transaction branch.
- static **const** int **TMSUSPEND**

  Flag that indicates that the caller is suspending (not terminating) its association with a transaction branch.
- static **const** int **XA_RDONLY**

  Flag that indicates that transaction work has been read only and has been committed normally.
- static **const** int **XA_OK**

  Flag that indicates that transaction work has been Prepared normally.

6.604.1 Detailed Description

The **XAResource** (p. 2269) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).

The XA interface defines the contract between a Resource Manager and a Transaction Manager in a distributed transaction processing (DTP) environment. A CMS provider implements this interface to support the association between a global transaction and a message broker connection.

The **XAResource** (p. 2269) is exposed to CMS client so that they can proxy calls from the Transaction Manager API of their choosing to the CMS provider. The CMS provider should behave and a standard XA Resource Manager its up to the client however to transmit the Transaction Manager's calls to the CMS provider through this interface.

Since

2.3

6.604.2 Constructor & Destructor Documentation

6.604.2.1 virtual cms::XAResource::~XAResource ( ) [virtual]

6.604.3 Member Function Documentation

6.604.3.1 virtual void cms::XAResource::commit (const Xid * *xid*, bool *onePhase*) [pure virtual]

Commits a global transaction.

Parameters

xid	the XID which identifies the global transaction.
onePhase	true if the resource manager should use a one-phase commit protocol to commit the transaction.

Exceptions

XAException *(p. 2265)*	if an error occurred.

Possible errors are identified by the errorcode in the **XAException** (p. 2265) and include: XA_HEURHAZ, XA_HEURCOM, XA_HEURRB, XA_HEURMIX, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO. In addition, one of the XA_RB∗ errors can occur if the transaction was not committed and onePhase was set to true. On completion of this method, the Resource Manager has rolled back the transaction and released resources held by the transaction.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 330).

6.604.3.2 virtual void cms::XAResource::end (const Xid ∗ *xid,* int *flags*) [pure virtual]

Ends the work done for a transaction branch.

The Resource Manager disconnects the XA resource from the transaction branch and allows the transaction to complete.

Parameters

xid	the XID which identifies the global transaction. Should have previously been used as the parameter to a start. method.
flags	a flags integer - one of: **XAResource::TMSUCCESS** (p. 2275), **XAResource::TMFAIL** (p. 2275), or **XAResource::TMSUSPEND** (p. 2275).

TMSUCCESS means that this section of work completed successfully.

TMFAIL means that this section of work failed. The Resource Manager can mark the transaction for rollback only.

TMSUSPEND means that this section of work is suspended and not yet complete. The associated transaction context is also suspended and must be restarted with a call to **start(Xid, int)** (p. **??**) with the TMRESUME flag.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, XAER_PROTO, or XA_RB∗.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 330).

6.604.3.3 virtual void cms::XAResource::forget (const Xid ∗ *xid*) [pure virtual]

Informs the Resource Manager that it can forget about a specified transaction branch.

Parameters

xid	the XID which identifies the global transaction.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 331).

6.604.3.4 virtual int cms::XAResource::getTransactionTimeout ( ) const [pure virtual]

Gets the transaction timeout value for this **XAResource** (p. 2269).

The default timeout value is the default timeout value set for the Resource Manager.

Returns

the transaction timeout value for this **XAResource** (p. 2269) in seconds.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 331).

6.604.3.5 virtual bool cms::XAResource::isSameRM (const XAResource * *theXAResource*) [pure virtual]

Returns true if the ResourceManager for this **XAResource** (p. 2269) is the same as the Resource Manager for a supplied **XAResource** (p. 2269).

Parameters

theXAResource	an **XAResource** (p. 2269) object

Returns

true if the Resource Manager for this **XAResource** (p. 2269) is the same as the Resource Manager for `theXAResource`.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR and XAER_RMFAIL.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 332).

6.604.3.6 virtual int cms::XAResource::prepare (const Xid * *xid*) [pure virtual]

Requests the Resource manager to prepare to commit a specified transaction.

Parameters

xid	the XID which identifies the global transaction.

Returns

an integer: XA_RDONLY or XA_OK. XA_OK implies that the transaction work has been prepared normally, XA_RDONLY implies that the transaction branch is read only and has been committed. If there is a failure which requires a rollback, an **XAException** (p. 2265) is raised.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XA_RB*, XAER_RMERR, XAER_RMFAIL, XAER_NOTA, XAER_INVAL, or XAER_PROTO.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 332).

6.604.3.7 virtual int cms::XAResource::recover (int *flag,* Xid ** *recovered*) [pure virtual]

Get a list of prepared transaction branches.

Typically used by a transaction manager during recovery to find transaction branches that are in prepared or heuristically completed states.

Parameters

flag	an integer. Must be one of: **XAResource::TMSTARTRSCAN** (p. 2275), **XAResource::TMENDRSCAN** (p. 2274), **XAResource::TMNOFLAGS** (p. 2275).

Returns

an array of zero or more XIDs identifying the transaction branches in the prepared or heuristically completed states.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, XAER_INVAL, and XAER_PROTO.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 333).

6.604.3.8 virtual void cms::XAResource::rollback (const Xid * *xid*) [pure virtual]

Requests the Resource Manager to rollback a specified transaction branch.

Parameters

xid	the XID which identifies the transaction branch.

Exceptions

XAException *(p. 2265)*	if an error occurs.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 333).

6.604.3.9 virtual bool cms::XAResource::setTransactionTimeout (int *seconds*) [pure virtual]

Sets the transaction timeout value for this **XAResource** (p. 2269).

If the value is set to 0, the default timeout value for the Resource Manager is used.

Parameters

seconds	the new Timeout value in seconds.

Returns

true if the transaction timeout value has been updated, false otherwise.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XAER_RMERR, XAER_RMFAIL, or XAER_INVAL.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 334).

6.604.3.10 virtual void cms::XAResource::start (const Xid * *xid,* int *flags*) [pure virtual]

Starts work for a specified transaction branch.

Parameters

xid	the XID which identifies the transaction branch.
flags	an integer. Must be one of **XAResource::TMNOFLAGS** (p. 2275), **XAResource::TMJOIN** (p. 2275), or **XAResource::TMRESUME** (p. 2275).

TMJOIN implies that the start applies to joining a transaction previously passed to the Resource Manager.

TMRESUME implies that the start applies to a suspended transaction that should be restarted.

If TMNOFLAGS is specified, then if the transaction has been previously seen by the Resource Manager, an **XAException** (p. 2265) is raised with the code XAER_DUPID.

Exceptions

XAException *(p. 2265)*	if an error occurs. Possible error identified in the errorcode include: XA_RB*, XAER_RMERR, XAER_RMFAIL, XAER_DUPID, XAER_OUTSIDE, XAER_NOTA, XAER_INVAL, or XAER_PROTO.

Implemented in **activemq::core::ActiveMQTransactionContext** (p. 334).

6.604.4 Field Documentation

6.604.4.1 const int cms::XAResource::TMENDRSCAN [static]

Flag to end a recovery scan.

6.604.4.2 const int cms::XAResource::TMFAIL [static]

Flag to indicate that the caller is dissociation from a transaction branch and that it should be marked rollback only.

6.604.4.3 const int cms::XAResource::TMJOIN [static]

Flag to indicate that the caller is joining sn existing transaction branch.

6.604.4.4 const int cms::XAResource::TMNOFLAGS [static]

Flag that indicates that no flags options are selected.

(ie a null flag)

6.604.4.5 const int cms::XAResource::TMONEPHASE [static]

Flag that indicates the caller is using one-phase commit optimization.

6.604.4.6 const int cms::XAResource::TMRESUME [static]

Flag that indicates the caller is resuming association with a suspended transaction branch.

6.604.4.7 const int cms::XAResource::TMSTARTRSCAN [static]

Flag that indicates the start of a recovery scan.

6.604.4.8 const int cms::XAResource::TMSUCCESS [static]

Flag that indicates the caller is dissociating from a transaction branch.

6.604.4.9 const int cms::XAResource::TMSUSPEND [static]

Flag that indicates that the caller is suspending (not terminating) its association with a transaction branch.

6.604.4.10 const int cms::XAResource::XA_OK [static]

Flag that indicates that transaction work has been Prepared normally.

6.604.4.11 const int cms::XAResource::XA_RDONLY [static]

Flag that indicates that transaction work has been read only and has been committed normally.

The documentation for this class was generated from the following file:

- src/main/cms/**XAResource.h**

6.605 cms::XASession Class Reference

The **XASession** (p. 2275) interface extends the capability of **Session** (p. 1830) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).

`#include <src/main/cms/XASession.h>`

Inheritance diagram for cms::XASession:

Public Member Functions

- virtual ~**XASession** () throw ()
- virtual **XAResource** * **getXAResource** () **const** =0

  *Returns the XA resource associated with this **Session** (p. 1830) to the caller.*

6.605.1 Detailed Description

The **XASession** (p. 2275) interface extends the capability of **Session** (p. 1830) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).

This support takes the form of a **cms::XAResource** (p. 2269) object. The functionality of this object closely resembles that defined by the standard X/Open XA Resource interface.

An application controls the transactional assignment of an **XASession** (p. 2275) by obtaining its **XAResource** (p. 2269). It uses the **XAResource** (p. 2269) to assign the session to a transaction, prepare and commit work on the transaction, and so on.

An **XAResource** (p. 2269) provides some fairly sophisticated facilities for interleaving work on multiple transactions, recovering a list of transactions in progress, and so on. A XA aware CMS provider must fully implement this functionality.

The **XASession** (p. 2275) instance will behave much like a normal **cms::Session** (p. 1830) however some methods will will not operate as normal, any call to **Session::commit** (p. 1834), or **Session::rollback** (p. 1841) will result in a **CMSException** (p. 640) being thrown. Also when not inside an XA transaction the **MessageConsumer** (p. 1455) will operate as if it were in the AutoAcknowlege mode.

The **XASession** (p. 2275) interface is optional. CMS providers are not required to support this interface. This interface is for use by CMS providers to support transactional environments. Client programs are strongly encouraged to use the transactional support available in their environment, rather than use these XA interfaces directly.

Since

2.3

6.605.2 Constructor & Destructor Documentation

6.605.2.1 virtual cms::XASession::~XASession ( ) throw () [virtual]

6.605.3 Member Function Documentation

6.605.3.1 virtual XAResource* cms::XASession::getXAResource ( ) const [pure virtual]

Returns the XA resource associated with this **Session** (p. 1830) to the caller.

The client can use the provided XA resource to interact with the XA Transaction Manager in use in the client application.

Returns

an XAResouce instance to the caller, the caller does not own this pointer and should not delete it.

Implemented in **activemq::core::ActiveMQXASession** (p. 339).

The documentation for this class was generated from the following file:

- src/main/cms/**XASession.h**

6.606 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)
- **XATransactionId** (**const cms::Xid** ∗xid)
- virtual ∼**XATransactionId** ()
- virtual unsigned char **getDataStructureType** () **const**

  *Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.*
- virtual **XATransactionId** ∗ **cloneDataStructure** () **const**

  Clone this obbject 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. 877) such as its type and value of its elements.*
- virtual bool **equals** (**const DataStructure** ∗value) **const**

  *Compares the **DataStructure** (p. 877) passed in to this one, and returns if they are equivalent.*
- virtual bool **isXATransactionId** () **const**
- virtual Xid ∗ **clone** () **const**

  Creates a Copy of this Xid instance that contains the same id values.
- virtual bool **equals** (**const** Xid ∗other) **const**
- virtual int **getBranchQualifier** (unsigned char ∗buffer, int size) **const**

  Gets the transaction branch qualifier component of the XID.
- virtual int **getGlobalTransactionId** (unsigned char ∗buffer, int size) **const**

  Gets the global transaction id component of the XID.
- virtual int **getFormatId** () **const**

  Gets the format identifier component of the XID.
- 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**

6.606.1 Member Typedef Documentation

6.606.1.1 typedef decaf::lang::PointerComparator<XATransactionId> activemq::commands::XATransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 2143).

6.606.2 Constructor & Destructor Documentation

6.606.2.1 activemq::commands::XATransactionId::XATransactionId ( )

6.606.2.2 activemq::commands::XATransactionId::XATransactionId (const XATransactionId & *other*)

6.606.2.3 activemq::commands::XATransactionId::XATransactionId (const cms::Xid * *xid*)

6.606.2.4 virtual activemq::commands::XATransactionId::~XATransactionId ( ) [virtual]

6.606.3 Member Function Documentation

6.606.3.1 virtual Xid* activemq::commands::XATransactionId::clone ( ) const [virtual]

Creates a Copy of this Xid instance that contains the same id values.

Returns

a new Xid instance that is equal to this one when compared.

Implements **cms::Xid** (p. 2285).

6.606.3.2 virtual XATransactionId* activemq::commands::XATransactionId::cloneDataStructure ( ) const [virtual]

Clone this obbject 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::TransactionId** (p. 2144).

6.606.3.3 virtual int activemq::commands::XATransactionId::compareTo (const XATransactionId & *value*) const [virtual]

6.606.3.4 virtual void activemq::commands::XATransactionId::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::TransactionId** (p. 2144).

6.606.3.5 virtual bool activemq::commands::XATransactionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 877) 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** (p. 877)'s are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 2144).

6.606.3.6 virtual bool activemq::commands::XATransactionId::equals (const Xid * *other*) const [virtual]

6.606.3.7 virtual bool activemq::commands::XATransactionId::equals (const XATransactionId & *value*) const [virtual]

6.606.3.8 virtual int activemq::commands::XATransactionId::getBranchQualifier (unsigned char * *buffer,* int *size*) const [virtual]

Gets the transaction branch qualifier component of the XID.

The value of this Xid's branch qualifier is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the branch qualifier then no copy is performed and the method returns -1.

Parameters

buffer	The location in memory to copy the qualifier bytes to.
size	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

XAException	if the size parameter is less than zero or buffer is NULL.

Implements **cms::Xid** (p. 2286).

6.606.3.9 virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ( ) const [virtual]

6.606.3.10 virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ( ) [virtual]

6.606.3.11 virtual unsigned char activemq::commands::XATransactionId::getDataStructureType ( ) const [virtual]

Get the **DataStructure** (p. 877) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Reimplemented from **activemq::commands::TransactionId** (p. 2145).

6.606.3.12 virtual int activemq::commands::XATransactionId::getFormatId ( ) const [virtual]

Gets the format identifier component of the XID.

Returns

an integer containing the format identifier. 0 means the OSI CCR format.

Implements **cms::Xid** (p. 2286).

6.606.3.13 virtual int activemq::commands::XATransactionId::getGlobalTransactionId (unsigned char * *buffer*, int *size*) const [virtual]

Gets the global transaction id component of the XID.

The value of this Xid's transaction id is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the transaction id then no copy is performed and the method returns -1.

Parameters

buffer	The location in memory to copy the transaction id bytes to.
size	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

XAException	if the size parameter is less than zero or buffer is NULL.

Implements **cms::Xid** (p. 2286).

6.606.3.14 virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId ( ) const [virtual]

6.606.3.15 virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId ( ) [virtual]

6.606.3.16 virtual bool activemq::commands::XATransactionId::isXATransactionId ( ) const [inline, virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2145).

6.606.3.17 virtual bool activemq::commands::XATransactionId::operator< (const XATransactionId & *value*) const [virtual]

6.606.3.18 XATransactionId& activemq::commands::XATransactionId::operator= (const XATransactionId & *other*)

6.606.3.19 virtual bool activemq::commands::XATransactionId::operator== (const XATransactionId & *value*) const [virtual]

6.606.3.20 virtual void activemq::commands::XATransactionId::setBranchQualifier (const std::vector< unsigned char > & *branchQualifier*) [virtual]

6.606.3.21 virtual void **activemq::commands::XATransactionId::setFormatId** (int *formatId*) [virtual]

6.606.3.22 virtual void **activemq::commands::XATransactionId::setGlobalTransactionId** (const std::vector< unsigned char > & *globalTransactionId*) [virtual]

6.606.3.23 virtual std::string **activemq::commands::XATransactionId::toString** ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 877) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 2145).

6.606.4 Field Documentation

6.606.4.1 std::vector<unsigned char> **activemq::commands::XATransactionId::branchQualifier** [protected]

6.606.4.2 int **activemq::commands::XATransactionId::formatId** [protected]

6.606.4.3 std::vector<unsigned char> **activemq::commands::XATransactionId::globalTransactionId** [protected]

6.606.4.4 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**

6.607 activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2281).

`#include <src/main/activemq/wireformat/openwire/marshal/generated/XATransaction-IdMarshaller.h>`

Inheritance diagram for activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual ~**XATransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () **const**

  Creates a new instance of the class that this class is a marshaling director for.
- virtual unsigned char **getDataStructureType** () **const**

  Gets the DataStructureType that this class marshals/unmarshals.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs)

Tight Un-marhsal to the given stream.

- virtual int **tightMarshal1** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut, **utils::BooleanStream** ∗bs)

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataInputStream** ∗dataIn)

  Loose Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** ∗wireFormat, **commands::DataStructure** ∗dataStructure, **decaf::io::DataOutputStream** ∗dataOut)

  Tight Marhsal to the given stream.

6.607.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2281).

NOTE!: This file is 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::generated::XATransactionIdMarshaller::XATransactionIdMarshaller ( ) [inline]

6.607.2.2 virtual activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::∼XATransactionIdMarshaller ( ) [inline, virtual]

6.607.3 Member Function Documentation

6.607.3.1 virtual commands::DataStructure∗ activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 869).

6.607.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 870).

6.607.3.3 virtual void activemq::wireformat::openwire::marshal::generated::XATransactionId-Marshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2146).

6.607.3.4 virtual void activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller-::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2147).

6.607.3.5 virtual int activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::tight-Marshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal to the given stream.

Parameters

format	The OpenwireFormat properties
command	The object to Marshal
bs	The boolean stream to marshal to.

Exceptions

IOException	if an error occurs.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2147).

6.607.3.6 virtual void activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) [virtual]

Tight Marhsal 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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2148).

6.607.3.7 virtual void activemq::wireformat::openwire::marshal::generated::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) [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.

Reimplemented from **activemq::wireformat::openwire::marshal::generated::TransactionIdMarshaller** (p. 2148).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/generated/**XATransactionIdMarshaller.h**

6.608 cms::Xid Class Reference

An interface which provides a mapping for the X/Open XID transaction identifier structure.

`#include <src/main/cms/Xid.h>`

Inheritance diagram for cms::Xid:

Public Member Functions

- **Xid** ()
- virtual ∼**Xid** ()
- virtual **Xid** ∗ **clone** () **const** =0

  *Creates a Copy of this **Xid** (p. 2285) instance that contains the same id values.*
- virtual bool **equals** (**const Xid** ∗other) **const** =0

  *Compares this **Xid** (p. 2285) to another and returns true if they are the same.*
- virtual int **getBranchQualifier** (unsigned char ∗buffer, int size) **const** =0

  Gets the transaction branch qualifier component of the XID.
- virtual int **getFormatId** () **const** =0

  Gets the format identifier component of the XID.
- virtual int **getGlobalTransactionId** (unsigned char ∗buffer, int size) **const** =0

  Gets the global transaction id component of the XID.

Static Public Attributes

- static **const** int **MAXGTRIDSIZE**

  The maximum number of bytes which will be copied into the array passed to getGlobaltransactionId().
- static **const** int **MAXBQUALSIZE**

  *The maximum number of bytes which will be copied into the array that is passed to **getBranchQualifier()** (p. 2286).*

6.608.1 Detailed Description

An interface which provides a mapping for the X/Open XID transaction identifier structure.

The **Xid** (p. 2285) interface is used by the Transaction Manager and the Resource managers. It is not typically used by application programs directly but the application developer must define a mechanism to map the calls and structures used by the Transaction Manager API in use into the format used by the CMS XA interfaces.

Since

2.3

6.608.2 Constructor & Destructor Documentation

6.608.2.1 cms::Xid::Xid ( )

6.608.2.2 virtual cms::Xid::∼Xid ( ) [virtual]

6.608.3 Member Function Documentation

6.608.3.1 virtual Xid∗ cms::Xid::clone ( ) const [pure virtual]

Creates a Copy of this **Xid** (p. 2285) instance that contains the same id values.

Returns

a new **Xid** (p. 2285) instance that is equal to this one when compared.

Implemented in **activemq::commands::XATransactionId** (p. 2278).

6.608.3.2 virtual bool cms::Xid::equals (const Xid * *other*) const [pure virtual]

Compares this **Xid** (p. 2285) to another and returns true if they are the same.

Returns

true if both **Xid** (p. 2285)'s represent that same id value.

6.608.3.3 virtual int cms::Xid::getBranchQualifier (unsigned char * *buffer,* int *size*) const [pure virtual]

Gets the transaction branch qualifier component of the XID.

The value of this **Xid** (p. 2285)'s branch qualifier is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the branch qualifier then no copy is performed and the method returns -1.

Parameters

buffer	The location in memory to copy the qualifier bytes to.
size	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

XAException *(p. 2265)*	if the size parameter is less than zero or buffer is NULL.

Implemented in **activemq::commands::XATransactionId** (p. 2279).

6.608.3.4 virtual int cms::Xid::getFormatId ( ) const [pure virtual]

Gets the format identifier component of the XID.

Returns

an integer containing the format identifier. 0 means the OSI CCR format.

Implemented in **activemq::commands::XATransactionId** (p. 2280).

6.608.3.5 virtual int cms::Xid::getGlobalTransactionId (unsigned char * *buffer,* int *size*) const [pure virtual]

Gets the global transaction id component of the XID.

The value of this **Xid** (p. 2285)'s transaction id is copied into the buffer provided. If the size of the provided buffer is not large enough to accommodate the transaction id then no copy is performed and the method returns -1.

Parameters

buffer	The location in memory to copy the transaction id bytes to.
size	The size of the buffer provided.

Returns

the number of bytes copied into the buffer, or -1 if the buffer provided was not large enough.

Exceptions

XAException *(p. 2265)*	if the size parameter is less than zero or buffer is NULL.

Implemented in **activemq::commands::XATransactionId** (p. 2280).

6.608.4 Field Documentation

6.608.4.1 const int cms::Xid::MAXBQUALSIZE [static]

The maximum number of bytes which will be copied into the array that is passed to **getBranchQualifier()** (p. 2286).

6.608.4.2 const int cms::Xid::MAXGTRIDSIZE [static]

The maximum number of bytes which will be copied into the array passed to getGlobaltransactionId().

The documentation for this class was generated from the following file:

- src/main/cms/**Xid.h**

6.609 decaf::util::logging::XMLFormatter Class Reference

Format a **LogRecord** (p. 1333) into a standard XML format.

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

Inheritance diagram for decaf::util::logging::XMLFormatter:

Public Member Functions

- **XMLFormatter** ()
- virtual ~**XMLFormatter** ()
- virtual std::string **format** (**const LogRecord** &record) **const**

  *Converts a **LogRecord** (p. 1333) into an XML string.*
- virtual std::string **getHead** (**const Handler** *handler)

  Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.
- virtual std::string **getTail** (**const Handler** *handler)

  Returns the tail string for a set of log records formatted as XML strings.

6.609.1 Detailed Description

Format a **LogRecord** (p. 1333) into a standard XML format.

TODO - Currently only outputs UTF-8 The **XMLFormatter** (p. 2287) can be used with arbitrary character encodings, but it is recommended that it normally be used with UTF-8. The character encoding can be set on the output **Handler** (p. 1085).

Since

1.0

6.609.2 Constructor & Destructor Documentation

6.609.2.1 decaf::util::logging::XMLFormatter::XMLFormatter ( )

6.609.2.2 virtual decaf::util::logging::XMLFormatter::~XMLFormatter ( ) [virtual]

6.609.3 Member Function Documentation

6.609.3.1 virtual std::string decaf::util::logging::XMLFormatter::format (const LogRecord & *record*) const [virtual]

Converts a **LogRecord** (p. 1333) into an XML string.

Parameters

record	The log record to be formatted.

Returns

the log record formatted as an XML string.

Implements **decaf::util::logging::Formatter** (p. 1074).

6.609.3.2 virtual std::string decaf::util::logging::XMLFormatter::getHead (const Handler * *handler*) [virtual]

Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.

Parameters

handler	The output handler, may be NULL.

Returns

the header string for log records formatted as XML strings.

6.609.3.3 virtual std::string decaf::util::logging::XMLFormatter::getTail (const Handler * *handler*) [virtual]

Returns the tail string for a set of log records formatted as XML strings.

Parameters

handler	The output handler, may be NULL.

Returns

the tail string for log records formatted as XML strings.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**XMLFormatter.h**

6.610 z_stream_s Struct Reference

`#include <src/main/decaf/internal/util/zip/zlib.h>`

Data Fields

- **Bytef ∗ next_in**
- **uInt avail_in**
- **uLong total_in**
- **Bytef ∗ next_out**
- **uInt avail_out**
- **uLong total_out**
- char ∗ **msg**
- struct **internal_state FAR ∗ state**
- alloc_func **zalloc**
- free_func **zfree**
- **voidpf opaque**
- int **data_type**
- **uLong adler**
- **uLong reserved**

6.610.1 Field Documentation

6.610.1.1 uLong z_stream_s::adler

6.610.1.2 uInt z_stream_s::avail_in

6.610.1.3 uInt z_stream_s::avail_out

6.610.1.4 int z_stream_s::data_type

6.610.1.5 char∗ z_stream_s::msg

6.610.1.6 Bytef∗ z_stream_s::next_in

6.610.1.7 Bytef∗ z_stream_s::next_out

6.610.1.8 voidpf z_stream_s::opaque

6.610.1.9 uLong z_stream_s::reserved

6.610.1.10 struct internal_state FAR∗ z_stream_s::state

6.610.1.11 uLong z_stream_s::total_in

6.610.1.12 uLong z_stream_s::total_out

6.610.1.13 alloc_func z_stream_s::zalloc

6.610.1.14 free_func z_stream_s::zfree

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**zlib.h**

6.611 decaf::util::zip::ZipException Class Reference

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

Inheritance diagram for decaf::util::zip::ZipException:

Public Member Functions

- **ZipException** () throw ()
  Default Constructor.
- **ZipException** (**const lang::Exception** &ex) throw ()
  Copy Constructor.
- **ZipException** (**const ZipException** &ex) throw ()
  Copy Constructor.
- **ZipException** (**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.
- **ZipException** (**const** std::exception ∗**cause**) throw ()
  Constructor.
- **ZipException** (**const** char ∗file, **const** int lineNumber, **const** char ∗msg,...) throw ()
  Constructor.
- virtual **ZipException** ∗ **clone** () **const**
  Clones this exception.
- virtual ∼**ZipException** () throw ()

6.611.1 Constructor & Destructor Documentation

6.611.1.1 decaf::util::zip::ZipException::ZipException ( ) throw ()

Default Constructor.

6.611.1.2 decaf::util::zip::ZipException::ZipException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy

6.611.1.3 decaf::util::zip::ZipException::ZipException (const ZipException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex	the exception to copy, which is an instance of this type

6.611.1.4 decaf::util::zip::ZipException::ZipException (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

6.611.1.5 decaf::util::zip::ZipException::ZipException (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.

6.611.1.6 decaf::util::zip::ZipException::ZipException (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

6.611.1.7 virtual decaf::util::zip::ZipException::~ZipException ( ) throw () [virtual]

6.611.2 Member Function Documentation

6.611.2.1 virtual ZipException* decaf::util::zip::ZipException::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. 1200).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**ZipException.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>
```

Data Structures

- class **activemq::cmsutil::CmsAccessor**

  *Base class for **activemq.cmsutil.CmsTemplate*** (p. *649*) *and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory*** (p. *741*) *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. *635*) *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 <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

Data Structures

- class **activemq::cmsutil::CmsTemplate**

  CmsTemplate (p. 649) *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. *649)**.*

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>
#include <decaf/util/LinkedList.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/decaf/internal/util/ResourceLifecycleManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/StlSet.h>
#include <decaf/internal/util/Resource.h>
```

Data Structures

- class **decaf::internal::util::ResourceLifecycleManager**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.13 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**

7.14 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**

7.15 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**

7.16 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**

7.17 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**

7.18 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**

7.19 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**

7.20 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/core/ActiveMQConnection.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>
#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**

7.21 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**

7.22 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**

7.23 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**

7.24 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**

7.25 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**

7.26 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**

7.27 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**

7.28 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**

7.29 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**

7.30 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**

7.31 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**

7.32 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**

7.33 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**

7.34 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**

7.35 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**

7.36 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**

7.37 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**

7.38 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**

7.39 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/commands/RemoveInfo.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**

7.40 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#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**

7.41 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**

7.42 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/commands/RemoveInfo.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**

7.43 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**

7.44 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**

7.45 src/main/activemq/commands/DataResponse.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::DataResponse**

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/DataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

Data Structures

- class **activemq::commands::DataStructure**

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/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**

7.48 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**

7.49 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**

7.50 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**

7.51 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**

7.52 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**

7.53 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**

7.54 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**

7.55 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**

7.56 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**

7.57 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**

7.58 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**

7.59 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/DataStructure.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**

7.60 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.

7.61 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**

7.62 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**

7.63 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**

7.64 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ProducerInfo.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**

7.65 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**

7.66 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**

7.67 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**

7.68 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**

7.69 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**

7.70 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/commands/RemoveInfo.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**

7.71 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**

7.72 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**

7.73 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**

7.74 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**

7.75 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**

7.76 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/RemoveInfo.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**

7.77 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**

7.78 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**

7.79 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**

7.80 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**

7.81 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**

7.82 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <cms/Xid.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**

7.83 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**

7.84 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/SessionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/threads/Scheduler.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/CopyOnWriteArrayList.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**

7.85 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.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**

7.86 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. 145) class.*

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/Dispatcher.h>
#include <activemq/core/RedeliveryPolicy.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
```

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/ActiveMQQueueBrowser.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/MessageEnumeration.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

Data Structures

- class **activemq::core::ActiveMQQueueBrowser**

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/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/Response.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 <activemq/threads/Scheduler.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/CopyOnWriteArrayList.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**

7.92 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**

7.93 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <cms/XAResource.h>
#include <cms/CMSException.h>
#include <cms/XAException.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**

7.94 src/main/activemq/core/ActiveMQXAConnection.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/XAConnection.h>
#include <activemq/core/ActiveMQConnection.h>
```

Data Structures

- class **activemq::core::ActiveMQXAConnection**

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/ActiveMQXAConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/XAConnectionFactory.h>
#include <activemq/core/ActiveMQConnectionFactory.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::core::ActiveMQXAConnectionFactory**

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/ActiveMQXASession.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/XASession.h>
#include <activemq/core/ActiveMQSession.h>
```

Data Structures

- class **activemq::core::ActiveMQXASession**

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/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 POCO 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**

7.98 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**

7.99 src/main/activemq/core/FifoMessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <decaf/util/LinkedList.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::FifoMessageDispatchChannel**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.100 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/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**

7.101 src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/PrefetchPolicy.h>
```

Data Structures

- class **activemq::core::policies::DefaultPrefetchPolicy**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**
- namespace **activemq::core::policies**

7.102 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/RedeliveryPolicy.h>
```

Data Structures

- class **activemq::core::policies::DefaultRedeliveryPolicy**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**
- namespace **activemq::core::policies**

7.103 src/main/activemq/core/PrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::core::PrefetchPolicy**

  Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.104 src/main/activemq/core/RedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::core::RedeliveryPolicy**

  *Interface for a **RedeliveryPolicy*** (p. *1744*) *object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.105 src/main/activemq/core/SimplePriorityMessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <decaf/util/LinkedList.h>
#include <decaf/lang/ArrayPointer.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::core::SimplePriorityMessageDispatchChannel**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.106 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. 2056), 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**

7.107 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**

7.108 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**

7.109 src/main/activemq/exceptions/ConnectionFailedException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::exceptions::ConnectionFailedException**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::exceptions**

7.110 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.

7.110.1 Define Documentation

7.110.1.1 #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().

7.110.1.2 #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).

7.110.1.3 #define AMQ_CATCH_RETHROW(*type*)

Value:

```
catch( type& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
        throw; \
    }
```

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().

7.110.1.4 #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.

7.110.1.5 #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().

7.111 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.

7.111.1 Define Documentation

7.111.1.1 #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().

7.111.1.2 #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).

7.111.1.3 #define DECAF_CATCH_RETHROW(*type*)

Value:

```
catch( type& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
        throw; \
    }
```

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(), and decaf::util::concurrent::LinkedBlockingQueue< E >::-LinkedBlockingQueue().

7.111.1.4 #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::util::ArrayList< E >::~ArrayList().

7.111.1.5 #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(), and decaf::util::concurrent::LinkedBlockingQueue< E >::-

LinkedBlockingQueue().

7.112 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**

7.113 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**

7.114 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**

7.115 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**

7.116 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. 675) 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**

7.117 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/commands/LocalTransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/util/LinkedList.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**

7.118 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**

7.119 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**

7.120 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**

7.121 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**

7.122 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**

7.123 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/LinkedList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.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**

7.124 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. *693).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.125 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/LinkedList.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. *2073) 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**

7.126 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**

7.127 src/main/activemq/threads/Scheduler.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/ServiceSupport.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/Timer.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <string>
```

Data Structures

- class **activemq::threads::Scheduler**

  Scheduler *(p. 1796) class for use in executing Runnable Tasks either periodically or one time only with optional delay.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.128 src/main/activemq/threads/SchedulerTimerTask.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
#include <decaf/lang/Runnable.h>
```

Data Structures

- class **activemq::threads::SchedulerTimerTask**

  Extension of the Decaf TimerTask that adds a Runnable instance which is the target of this task.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.129 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**

7.130 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**

7.131 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/util/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::AbstractTransportFactory**

  *Abstract implementation of the **TransportFactory** (p. 2167) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2167) instances.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.132 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. 2161) is a **Transport** (p. 2161) 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**

7.133 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**

7.134 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**

7.135 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**

  *A Utility class that create empty implementations for the **TransportListener** (p. 2176) interface so that a subclass only needs to override the one's its interested.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.136 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**

7.137 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/LinkedList.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**

7.138 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/LinkedList.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**

7.139 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ConnectionId.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/LinkedList.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**

7.140 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. *1010*).

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.141 src/main/activemq/transport/failover/FailoverTransportListener.h 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. *2161*) *to perform the work of responding to events from the active **Transport*** (p. *2161*).

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.142 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
#include <decaf/util/LinkedList.h>
#include <decaf/util/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**

7.143 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**

7.144 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**

7.145 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**

7.146 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. 2161) 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**

7.147 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.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**

7.148 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/LinkedList.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. *1509*).

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.149 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/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. 1509) defines a base level **Transport** (p. 2161) class that is intended to be used in place of an a regular protocol **Transport** (p. 2161) suck 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**

7.150 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**

7.151 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/LinkedList.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**

7.152 src/main/activemq/transport/tcp/SslTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransport.h>
```

Data Structures

- class **activemq::transport::tcp::SslTransport**

  Transport *(p. 2161) for connecting to a Broker using an SSL Socket.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.153 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransportFactory.h>
```

Data Structures

- class **activemq::transport::tcp::SslTransportFactory**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.154 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. 1200).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.155 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. 2086).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.156 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/util/List.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**

7.157 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**

7.158 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**

7.159 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**

7.160 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/util/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. *2161*) *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**

7.161 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. *1705*) *object.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.162 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/MessageNotWriteableException.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.

7.162.1 Define Documentation

7.162.1.1 #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 >::getByteProperty(), 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().

7.163 src/main/activemq/util/CompositeData.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/LinkedList.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**

7.164 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**

7.164.1 Define Documentation

7.164.1.1 #define AMQCPP_API

7.164.1.2 #define HAVE_PTHREAD_H

7.164.1.3 #define HAVE_UUID_T

7.164.1.4 #define HAVE_UUID_UUID_H

7.165 src/main/cms/Config.h File Reference

Defines

- #define **CMS_API**

7.165.1 Define Documentation

7.165.1.1 #define CMS_API

7.166 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**

7.166.1 Define Documentation

7.166.1.1 #define DECAF_API

7.166.1.2 #define DECAF_UNUSED

7.166.1.3 #define HAVE_PTHREAD_H

7.166.1.4 #define HAVE_UUID_T

7.166.1.5 #define HAVE_UUID_UUID_H

7.167 src/main/activemq/util/IdGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
```

Data Structures

- class **activemq::util::IdGenerator**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::library**
- namespace **activemq::util**

7.168 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**

7.169 src/main/activemq/util/MarshallingSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <string>
```

Data Structures

- class **activemq::util::MarshallingSupport**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.170 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**

7.171 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/LinkedList.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**

7.172 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/util/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**

7.173 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. 1662) 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**

7.174 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/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**

7.175 src/main/activemq/util/Service.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::Service**

  *Base interface for all classes that run as a **Service** (p. 1826) inside the application.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.176 src/main/activemq/util/ServiceListener.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::ServiceListener**

  *Listener interface for observers of **Service*** (p. *1826*) *related events.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.177 src/main/activemq/util/ServiceStopper.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **activemq::util::ServiceStopper**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.178 src/main/activemq/util/ServiceSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Service.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/CopyOnWriteArrayList.h>
```

Data Structures

- class **activemq::util::ServiceSupport**

  *Provides a base class for **Service** (p. 1826) implementations.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

Defines

- #define **SERVICESUPPORT_H_**

7.178.1 Define Documentation

7.178.1.1 #define SERVICESUPPORT_H_

7.179 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/LinkedList.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**

7.180 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**

7.181 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>
#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**

7.182 src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
#include <activemq/wireformat/openwire/utils/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 OpenWire 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**

7.183 src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h 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**

7.184 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQBlobMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 126).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.185 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQBytesMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 142).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.186 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 200).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.187 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQMapMessage-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQMapMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 220).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.188 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQMessage-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 225).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.189 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQObjectMessage-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQObjectMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 235).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.190 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/ActiveMQDestination-
Marshaller.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::generated::ActiveMQQueueMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 255).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.191 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQStream-MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQStreamMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 290).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.192 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestination-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/ActiveMQDestination-
```

```
Marshaller.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::generated::ActiveMQTempDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (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::generated**

7.193 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempQueue-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestination-
Marshaller.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::generated::ActiveMQTempQueueMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 303).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.194 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/ActiveMQTempDestination-
Marshaller.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::generated::ActiveMQTempTopicMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 310).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.195 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ActiveMQTextMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 318).*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.196 src/main/activemq/wireformat/openwire/marshal/generated/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/ActiveMQDestination-
Marshaller.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::generated::ActiveMQTopicMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 324).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.197 src/main/activemq/wireformat/openwire/marshal/generated/BaseCommandMarshaller.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::generated::BaseCommandMarshaller**

  *Marshaling code for Open Wire Format for **BaseCommandMarshaller** (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::generated**

7.198 src/main/activemq/wireformat/openwire/marshal/generated/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/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::BrokerIdMarshaller**

  *Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 440).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.199 src/main/activemq/wireformat/openwire/marshal/generated/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::BrokerInfoMarshaller**

  *Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 448).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.200 src/main/activemq/wireformat/openwire/marshal/generated/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ConnectionControlMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 732).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.201 src/main/activemq/wireformat/openwire/marshal/generated/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ConnectionErrorMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 738).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.202 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::ConnectionIdMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 748).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.203 src/main/activemq/wireformat/openwire/marshal/generated/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ConnectionInfoMarshaller**

  *Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 756).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.204 src/main/activemq/wireformat/openwire/marshal/generated/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ConsumerControlMarshaller**

  *Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 773).*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.205 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::ConsumerIdMarshaller**

  *Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 779).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.206 src/main/activemq/wireformat/openwire/marshal/generated/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ConsumerInfoMarshaller**

  *Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 789).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.207 src/main/activemq/wireformat/openwire/marshal/generated/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ControlCommandMarshaller**

  *Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 795).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.208 src/main/activemq/wireformat/openwire/marshal/generated/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::DataArrayResponseMarshaller**

  *Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 831).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.209 src/main/activemq/wireformat/openwire/marshal/generated/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::DataResponseMarshaller**

  *Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 865).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.210 src/main/activemq/wireformat/openwire/marshal/generated/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::DestinationInfoMarshaller**

  *Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 943).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.211 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::DiscoveryEventMarshaller**

  *Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 952).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.212 src/main/activemq/wireformat/openwire/marshal/generated/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ExceptionResponseMarshaller**

  *Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 999).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.213 src/main/activemq/wireformat/openwire/marshal/generated/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::FlushCommandMarshaller**

  *Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1070).*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.214 src/main/activemq/wireformat/openwire/marshal/generated/IntegerResponse-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::IntegerResponseMarshaller**

  *Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1177).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.215 src/main/activemq/wireformat/openwire/marshal/generated/JournalQueueAck-Marshaller.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::generated::JournalQueueAckMarshaller**

  *Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1213).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.216 src/main/activemq/wireformat/openwire/marshal/generated/JournalTopicAck-Marshaller.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::generated::JournalTopicAckMarshaller**

  *Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1219).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.217 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::JournalTraceMarshaller**

  *Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1224).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.218 src/main/activemq/wireformat/openwire/marshal/generated/JournalTransaction-Marshaller.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::generated::JournalTransactionMarshaller**

  *Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (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::generated**

7.219 src/main/activemq/wireformat/openwire/marshal/generated/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::KeepAliveInfoMarshaller**

  *Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1236).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.220 src/main/activemq/wireformat/openwire/marshal/generated/LastPartialCommand-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/PartialCommand-
Marshaller.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::generated::LastPartialCommandMarshaller**

  *Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1247).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.221 src/main/activemq/wireformat/openwire/marshal/generated/LocalTransactionId-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/TransactionId-
Marshaller.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::generated::LocalTransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1301).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.222 src/main/activemq/wireformat/openwire/marshal/generated/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::generated::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::generated**

7.223 src/main/activemq/wireformat/openwire/marshal/generated/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::MessageAckMarshaller**

  *Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1452).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.224 src/main/activemq/wireformat/openwire/marshal/generated/MessageDispatch-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::MessageDispatchMarshaller**

  *Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1466).*

Namespaces

- namespace **activemq**

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

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.225 src/main/activemq/wireformat/openwire/marshal/generated/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::MessageDispatchNotificationMarshaller**

  *Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1472).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.226 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::MessageIdMarshaller**

  *Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1482).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.227 src/main/activemq/wireformat/openwire/marshal/generated/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::MessageMarshaller**

  *Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1486).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.228 src/main/activemq/wireformat/openwire/marshal/generated/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::MessagePullMarshaller**

  *Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1506).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.229 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::NetworkBridgeFilterMarshaller**

  *Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1530).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.230 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::PartialCommandMarshaller**

  *Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 1611).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.231 src/main/activemq/wireformat/openwire/marshal/generated/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ProducerAckMarshaller**

  *Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 1686).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.232 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::ProducerIdMarshaller**

  *Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 1694).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.233 src/main/activemq/wireformat/openwire/marshal/generated/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ProducerInfoMarshaller**

  *Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 1701).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.234 src/main/activemq/wireformat/openwire/marshal/generated/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::RemoveInfoMarshaller**

  *Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 1761).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.235 src/main/activemq/wireformat/openwire/marshal/generated/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::RemoveSubscriptionInfoMarshaller**

  *Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 1767).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.236 src/main/activemq/wireformat/openwire/marshal/generated/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ReplayCommandMarshaller**

  *Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (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::generated**

7.237 src/main/activemq/wireformat/openwire/marshal/generated/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ResponseMarshaller**

  *Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 1788).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.238 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::SessionIdMarshaller**

  *Marshaling code for Open Wire Format for **SessionIdMarshaller** (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::generated**

7.239 src/main/activemq/wireformat/openwire/marshal/generated/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::SessionInfoMarshaller**

  *Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 1851).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.240 src/main/activemq/wireformat/openwire/marshal/generated/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::ShutdownInfoMarshaller**

  *Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 1886).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.241 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::SubscriptionInfoMarshaller**

  *Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2042).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.242 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::TransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **TransactionIdMarshaller** (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::generated**

7.243 src/main/activemq/wireformat/openwire/marshal/generated/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/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::generated::TransactionInfoMarshaller**

  *Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2152).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.244 src/main/activemq/wireformat/openwire/marshal/generated/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::generated::WireFormatInfoMarshaller**

  *Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2248).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.245 src/main/activemq/wireformat/openwire/marshal/generated/XATransactionId-Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/generated/TransactionId-
Marshaller.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::generated::XATransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2281).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::generated**

7.246 src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h 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**

7.247 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**

7.248 src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h 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**

7.249 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h 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**

7.250 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/LinkedList.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

  Used to allow a MockTransport to generate response commands to OpenWire Commands.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.251 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**

7.252 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. 1089) class maps hexidecimal 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::utils**

7.253 src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h 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::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.

Namespaces

- namespace **activemq**

  Licensed 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**

7.254 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**

7.255 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**

7.256 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** (p. 2005)'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**

7.257 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**

7.258 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**

7.259 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**

7.260 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. 2239) is the interface that all **WireFormatFactory** (p. 2239) classes must extend.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.261 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. 2251) which allows a **WireFormat** (p. 2236) to.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.262 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/util/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. 2236) 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**

7.263 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/MessageNotWriteableException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
```

Data Structures

- class **cms::BytesMessage**

  *A **BytesMessage** (p. 557) 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.

7.264 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.

7.265 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**

7.266 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.

7.267 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.

7.268 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.

7.269 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stoppable.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.

7.270 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. 725) objects returned implement the CMS **Connection** (p. 725) 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.

7.271 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ConnectionMetaData**

  *A **ConnectionMetaData** (p. 759) object provides information describing the **Connection** (p. 725) object.*

Namespaces

- namespace **cms**

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

7.272 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.

7.273 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. 936) object encapsulates a provider-specific address.*

Namespaces

- namespace **cms**

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

7.274 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. 996) that is registered with the **Connection** (p. 725).*

Namespaces

- namespace **cms**

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

7.275 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.

7.276 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**

7.277 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.

7.278 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.

7.279 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.

7.280 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. 1379) 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.

7.281 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
#include <cms/Startable.h>
#include <cms/Stoppable.h>
```

Data Structures

- class **cms::MessageConsumer**

  *A client uses a **MessageConsumer** (p. 1455) to received messages from a destination.*

Namespaces

- namespace **cms**

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

7.282 src/main/cms/MessageEnumeration.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageEnumeration**

  Defines an object that enumerates a collection of Messages.

Namespaces

- namespace **cms**

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

7.283 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageEOFException**

  *This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2020) or **BytesMessage** (p. 557) is being read.*

Namespaces

- namespace **cms**

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

7.284 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.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.

7.285 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::MessageListener**

  *A **MessageListener** (p. 1485) 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.

7.286 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.

7.287 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.

7.288 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. 1491)* *object to send messages to a* ***Destination*** *(p. 936)**.*

Namespaces

- namespace **cms**

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

7.289 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.

7.290 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.

7.291 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/IndexOutOfBoundsException.h>
#include <decaf/util/NoSuchElementException.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**

7.292 src/main/cms/QueueBrowser.h File Reference

```
#include <string>
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Queue.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageEnumeration.h>
```

Data Structures

- class **cms::QueueBrowser**

  *This class implements in interface for browsing the messages in a **Queue*** (p. 1722) *without removing them.*

Namespaces

- namespace **cms**

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

7.293 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Startable.h>
#include <cms/Stoppable.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. 1830) 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.

7.294 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.

7.295 src/main/cms/Stoppable.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.

7.296 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. *2020*).

Namespaces

- namespace **cms**

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

7.297 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. *1722*) *based **Destination*** (p. *936*).

Namespaces

- namespace **cms**

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

7.298 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. *2141*) *based **Destination*** (p. *936*).

Namespaces

- namespace **cms**

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

7.299 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.

7.300 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.

7.301 src/main/cms/TransactionInProgressException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TransactionInProgressException**

  This exception is thrown when an operation is invalid because a transaction is in progress.

Namespaces

- namespace **cms**

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

7.302 src/main/cms/TransactionRolledBackException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TransactionRolledBackException**

  This exception must be thrown when a call to ***Session.commit*** (p. *1834*) *results in a rollback of the current transaction.*

Namespaces

- namespace **cms**

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

7.303 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.

7.304 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**

7.305 src/main/cms/XAConnection.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
#include <cms/Connection.h>
#include <cms/XASession.h>
```

Data Structures

- class **cms::XAConnection**

  *The **XAConnection** (p. 2262) interface defines an extended **Connection** (p. 725) type that is used to create **XASession** (p. 2275) objects.*

Namespaces

- namespace **cms**

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

7.306 src/main/cms/XAConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/XAConnection.h>
#include <cms/XAException.h>
```

Data Structures

- class **cms::XAConnectionFactory**

  *The **XAConnectionFactory** (p. 2263) interface is specialized interface that defines an **ConnectionFactory** (p. 741) that creates **Connection** (p. 725) instance that will participate in XA Transactions.*

Namespaces

- namespace **cms**

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

7.307 src/main/cms/XAException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::XAException**

  *The **XAException** (p. 2265) is thrown by the Resource Manager (RM) to inform the Transaction Manager of an error encountered by the involved transaction.*

Namespaces

- namespace **cms**

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

7.308 src/main/cms/XAResource.h File Reference

```
#include <cms/Config.h>
#include <cms/Xid.h>
#include <cms/XAException.h>
```

Data Structures

- class **cms::XAResource**

  *The **XAResource** (p. 2269) interface is a CMS mapping of the industry standard XA interface based on the X/Open CAE Specification (Distributed Transaction Processing: The XA Specification).*

Namespaces

- namespace **cms**

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

7.309 src/main/cms/XASession.h File Reference

```
#include <cms/Config.h>
#include <cms/Session.h>
#include <cms/XAResource.h>
```

Data Structures

- class **cms::XASession**

  *The **XASession** (p. 2275) interface extends the capability of **Session** (p. 1830) by adding access to a CMS provider's support for the operating inside an XA Transaction (optional).*

Namespaces

- namespace **cms**

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

7.310 src/main/cms/Xid.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Xid**

  An interface which provides a mapping for the X/Open XID transaction identifier structure.

Namespaces

- namespace **cms**

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

7.311 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**

7.312 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runtime.h>
#include <decaf/util/concurrent/Mutex.h>
#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**

7.313 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.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**

7.314 src/main/decaf/internal/io/StandardInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.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**

7.315 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.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**

7.316 src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::DefaultServerSocketFactory**

  Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.317 src/main/decaf/internal/net/DefaultSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
```

Data Structures

- class **decaf::internal::net::DefaultSocketFactory**

  SocketFactory implementation that is used to create Sockets.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.318 src/main/decaf/internal/net/Network.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/internal/util/Resource.h>
#include <decaf/internal/util/GenericResource.h>
```

Data Structures

- class **decaf::internal::net::Network**

  Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.319 src/main/decaf/internal/net/SocketFileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FileDescriptor.h>
```

Data Structures

- class **decaf::internal::net::SocketFileDescriptor**

  File Descriptor type used internally by Decaf Socket objects.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.320 src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContext.h>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLContext**

  Default SSLContext manager for the Decaf library.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.321 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLServerSocketFactory**

  Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.322 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
#include <string>
#include <vector>
```

Data Structures

- class **decaf::internal::net::ssl::DefaultSSLSocketFactory**

  Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

7.323 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLContextSpi**

  Provides an SSLContext that wraps the OpenSSL API.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.324 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLParameters**

  Container class for parameters that are Common to OpenSSL socket classes.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.325 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocket.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocket**

  SSLServerSocket based on OpenSSL library code.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.326 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory**

  SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.327 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocket.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocket**

  Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.328 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketException**

  Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.329 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory**

  Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.330 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream**

  An output stream for reading data from an OpenSSL Socket instance.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.331 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream**

  *OutputStream implementation used to write data to an **OpenSSLSocket** (p. 1560) instance.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

7.332 src/main/decaf/internal/net/tcp/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
#include <decaf/io/IOException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::internal::net::tcp::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::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.333 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocketInputStream**

  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::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.334 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

Data Structures

- class **decaf::internal::net::tcp::TcpSocketOutputStream**

  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::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

7.335 src/main/decaf/internal/net/URIEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <string>
```

Data Structures

- class **decaf::internal::net::URIEncoderDecoder**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.336 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**

7.337 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**

7.338 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>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::internal::nio::BufferFactory**

  *Factory class used by static methods in the **decaf::nio** (p. 74) 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**

7.339 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/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.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>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::nio::ByteArrayBuffer**

  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**

7.340 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/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.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**

7.341 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/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.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**

7.342 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/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.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**

7.343 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/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.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**

7.344 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/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.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**

7.345 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/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.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**

7.346 src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

Data Structures

- class **decaf::internal::security::SecureRandomImpl**

  Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::security**

7.347 src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

Data Structures

- class **decaf::internal::security::SecureRandomImpl**

  Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::security**

7.348 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.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**

7.349 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**

7.350 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**

7.351 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**

7.352 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**

7.353 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**

7.354 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**

7.355 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**

7.356 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**

7.357 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**

7.358 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**

7.359 src/main/decaf/internal/util/GenericResource.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/internal/util/Resource.h>
```

Data Structures

- class **decaf::internal::util::GenericResource**< **T** >

  *A Generic **Resource** (p. 1777) wraps some type and will delete it when the **Resource** (p. 1777) itself is deleted.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.360 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**

7.361 src/main/decaf/internal/util/Resource.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::Resource**

  Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.362 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**

7.363 src/main/decaf/internal/util/zip/crc32.h File Reference

Variables

- **local const** unsigned long **FAR crc_table** [TBLS][256]

7.363.1 Variable Documentation

7.363.1.1 local const unsigned long FAR crc_table[TBLS][256]

7.364 src/main/decaf/internal/util/zip/deflate.h File Reference

```
#include "zutil.h"
```

Data Structures

- struct **ct_data_s**
- struct **tree_desc_s**
- struct **internal_state**

Defines

- #define **GZIP**
- #define **LENGTH_CODES** 29
- #define **LITERALS** 256
- #define **L_CODES** (**LITERALS**+1+**LENGTH_CODES**)
- #define **D_CODES** 30
- #define **BL_CODES** 19
- #define **HEAP_SIZE** (2∗**L_CODES**+1)
- #define **MAX_BITS** 15
- #define **INIT_STATE** 42
- #define **EXTRA_STATE** 69
- #define **NAME_STATE** 73
- #define **COMMENT_STATE** 91
- #define **HCRC_STATE** 103
- #define **BUSY_STATE** 113
- #define **FINISH_STATE** 666
- #define **Freq** fc.freq
- #define **Code** fc.code
- #define **Dad** dl.dad
- #define **Len** dl.len
- #define **max_insert_length** max_lazy_match
- #define **put_byte**(s, c) {s->pending_buf[s->pending++] = (c);}
- #define **MIN_LOOKAHEAD** (**MAX_MATCH**+**MIN_MATCH**+1)
- #define **MAX_DIST**(s) ((s)->w_size-**MIN_LOOKAHEAD**)
- #define **WIN_INIT MAX_MATCH**
- #define **d_code**(dist) ((dist) < 256 ? **_dist_code**[dist] : **_dist_code**[256+((dist)>>7)])
- #define **_tr_tally_lit**(s, c, flush)
- #define **_tr_tally_dist**(s, distance, length, flush)

Typedefs

- typedef struct **ct_data_s ct_data**
- typedef struct static_tree_desc_s **static_tree_desc**
- typedef struct **tree_desc_s tree_desc**
- typedef **ush Pos**
- typedef **Pos FAR Posf**
- typedef unsigned **IPos**
- typedef struct **internal_state deflate_state**

Functions

- void **ZLIB_INTERNAL** _tr_init **OF** ((**deflate_state** ∗s))
- int **ZLIB_INTERNAL** _tr_tally **OF** ((**deflate_state** ∗s, unsigned dist, unsigned lc))
- void **ZLIB_INTERNAL** _tr_flush_block **OF** ((**deflate_state** ∗s, **charf** ∗buf, **ulg** stored_len, int last))

Variables

- **uch ZLIB_INTERNAL _length_code** []
- **uch ZLIB_INTERNAL _dist_code** []

7.364.1 Define Documentation

7.364.1.1 #define _tr_tally_dist(*s, distance, length, flush*)

Value:

```
{ uch len = (length); \
    ush dist = (distance); \
    s->d_buf[s->last_lit] = dist; \
    s->l_buf[s->last_lit++] = len; \
    dist--; \
    s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
    s->dyn_dtree[d_code(dist)].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
  }
```

7.364.1.2 #define _tr_tally_lit(*s, c, flush*)

Value:

```
{ uch cc = (c); \
    s->d_buf[s->last_lit] = 0; \
    s->l_buf[s->last_lit++] = cc; \
    s->dyn_ltree[cc].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
   }
```

7.364.1.3 #define BL_CODES 19

7.364.1.4 #define BUSY_STATE 113

7.364.1.5 #define Code fc.code

7.364.1.6 #define COMMENT_STATE 91

7.364.1.7 #define d_code(*dist*) ((dist) < 256 ? _dist_code[dist] : _dist_code[256+((dist)>>7)])

7.364.1.8 #define D_CODES 30

7.364.1.9 #define Dad dl.dad

7.364.1.10 #define EXTRA_STATE 69

7.364.1.11 #define FINISH_STATE 666

7.364.1.12 #define Freq fc.freq

7.364.1.13 #define GZIP

7.364.1.14 #define HCRC_STATE 103

7.364.1.15 #define HEAP_SIZE (2*L_CODES+1)

7.364.1.16 #define INIT_STATE 42

7.364.1.17 #define L_CODES (LITERALS+1+LENGTH_CODES)

7.364.1.18 #define Len dl.len

7.364.1.19 #define LENGTH_CODES 29

7.364.1.20 #define LITERALS 256

7.364.1.21 #define MAX_BITS 15

7.364.1.22 #define MAX_DIST(*s*) ((s)->w_size-MIN_LOOKAHEAD)

7.364.1.23 #define max_insert_length max_lazy_match

7.364.1.24 #define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)

7.364.1.25 #define NAME_STATE 73

7.364.1.26 #define put_byte(*s, c*) {s->pending_buf[s->pending++] = (c);}

7.364.1.27 #define WIN_INIT MAX_MATCH

7.364.2 Typedef Documentation

7.364.2.1 typedef struct ct_data_s ct_data

7.364.2.2 typedef struct internal_state deflate_state

7.364.2.3 typedef unsigned IPos

7.364.2.4 typedef ush Pos

7.364.2.5 typedef Pos FAR Posf

7.364.2.6 typedef struct static_tree_desc_s static_tree_desc

7.364.2.7 typedef struct tree_desc_s tree_desc

7.364.3 Function Documentation

7.364.3.1 void ZLIB_INTERNAL _tr_init OF ((deflate_state *s))

7.364.3.2 int ZLIB_INTERNAL _tr_tally OF ((deflate_state *s, unsigned dist, unsigned lc))

7.364.3.3 void ZLIB_INTERNAL _tr_flush_block OF ((deflate_state *s, charf *buf, ulg stored_len, int last))

7.364.4 Variable Documentation

7.364.4.1 uch ZLIB_INTERNAL _dist_code[]

7.364.4.2 uch ZLIB_INTERNAL _length_code[]

7.365 src/main/decaf/internal/util/zip/gzguts.h File Reference

```
#include <stdio.h>
#include "zlib.h"
#include <fcntl.h>
```

Data Structures

- struct **gz_state**

Defines

- #define **ZLIB_INTERNAL**
- #define **local** static
- #define **zstrerror**() "stdio error (consult errno)"
- #define **GZBUFSIZE** 8192
- #define **GZ_NONE** 0
- #define **GZ_READ** 7247
- #define **GZ_WRITE** 31153
- #define **GZ_APPEND** 1 /∗ mode set to **GZ_WRITE** after the file is opened ∗/
- #define **LOOK** 0 /∗ look for a gzip header ∗/
- #define **COPY** 1 /∗ copy input directly ∗/
- #define **GZIP** 2 /∗ decompress a gzip stream ∗/
- #define **GT_OFF**(x) (sizeof(int) == sizeof(**z_off64_t**) && (x) > gz_intmax())

Typedefs

- typedef **gz_state FAR** ∗ **gz_statep**

Functions

- **voidp** malloc **OF** ((**uInt** size))
- void free **OF** ((**voidpf** ptr))
- **ZEXTERN gzFile ZEXPORT** gzopen64 **OF** ((**const** char ∗, **const** char ∗))
- **ZEXTERN z_off64_t ZEXPORT** gzseek64 **OF** ((**gzFile**, **z_off64_t**, int))
- **ZEXTERN z_off64_t ZEXPORT** gztell64 **OF** ((**gzFile**))
- void **ZLIB_INTERNAL** gz_error **OF** ((**gz_statep**, int, **const** char ∗))
- unsigned **ZLIB_INTERNAL** gz_intmax **OF** ((void))

7.365.1 Define Documentation

7.365.1.1 #define COPY 1 /∗ copy input directly ∗/

7.365.1.2 #define GT_OFF(*x*) (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())

7.365.1.3 #define GZ_APPEND 1 /∗ mode set to GZ_WRITE after the file is opened ∗/

7.365.1.4 #define GZ_NONE 0

7.365.1.5 #define GZ_READ 7247

7.365.1.6 #define GZ_WRITE 31153

7.365.1.7 #define GZBUFSIZE 8192

7.365.1.8 #define GZIP 2 /∗ decompress a gzip stream ∗/

7.365.1.9 #define local static

7.365.1.10 #define LOOK 0 /* look for a gzip header */

7.365.1.11 #define ZLIB_INTERNAL

7.365.1.12 #define zstrerror( ) "stdio error (consult errno)"

7.365.2 Typedef Documentation

7.365.2.1 typedef gz_state FAR* gz_statep

7.365.3 Function Documentation

7.365.3.1 voidp malloc OF ((uInt size))

7.365.3.2 void free OF ((voidpf ptr))

7.365.3.3 ZEXTERN gzFile ZEXPORT gzopen64 OF ((const char *, const char *))

7.365.3.4 ZEXTERN z_off64_t ZEXPORT gzseek64 OF ((gzFile, z_off64_t, int))

7.365.3.5 ZEXTERN z_off64_t ZEXPORT gztell64 OF ((gzFile))

7.365.3.6 void ZLIB_INTERNAL gz_error OF ((gz_statep, int, const char *))

7.365.3.7 unsigned ZLIB_INTERNAL gz_intmax OF ((void))

7.366 src/main/decaf/internal/util/zip/inffast.h File Reference

Functions

- void **ZLIB_INTERNAL** inflate_fast **OF** ((**z_streamp** strm, unsigned start))

7.366.1 Function Documentation

7.366.1.1 void ZLIB_INTERNAL inflate_fast OF ((z_streamp strm, unsigned start))

7.367 src/main/decaf/internal/util/zip/inffixed.h File Reference

7.368 src/main/decaf/internal/util/zip/inflate.h File Reference

Data Structures

- struct **inflate_state**

Defines

- #define **GUNZIP**

Enumerations

- enum **inflate_mode** {
  HEAD, **FLAGS**, **TIME**, **OS**,
  EXLEN, **EXTRA**, **NAME**, **COMMENT**,
  HCRC, **DICTID**, **DICT**, **TYPE**,
  TYPEDO, **STORED**, **COPY_**, **COPY**,
  TABLE, **LENLENS**, **CODELENS**, **LEN_**,
  LEN, **LENEXT**, **DIST**, **DISTEXT**,
  MATCH, **LIT**, **CHECK**, **LENGTH**,
  DONE, **BAD**, **MEM**, **SYNC** }

7.368.1 Define Documentation

7.368.1.1 #define GUNZIP

7.368.2 Enumeration Type Documentation

7.368.2.1 enum inflate_mode

Enumerator:

HEAD

FLAGS

TIME

OS

EXLEN

EXTRA

NAME

COMMENT

HCRC

DICTID

DICT

TYPE

TYPEDO

STORED

COPY_

COPY

TABLE

LENLENS

CODELENS

LEN_

LEN

LENEXT

DIST

DISTEXT

MATCH

LIT

CHECK

LENGTH

DONE

BAD

MEM

SYNC

7.369 src/main/decaf/internal/util/zip/inftrees.h File Reference

Data Structures

- struct **code**

Defines

- #define **ENOUGH_LENS** 852
- #define **ENOUGH_DISTS** 592
- #define **ENOUGH** (**ENOUGH_LENS**+**ENOUGH_DISTS**)

Enumerations

- enum **codetype** { **CODES**, **LENS**, **DISTS** }

Functions

- int **ZLIB_INTERNAL** inflate_table **OF** ((**codetype** type, unsigned short **FAR** ∗lens, unsigned codes, **code** **FAR** ∗**FAR** ∗table, unsigned **FAR** ∗bits, unsigned short **FAR** ∗work))

7.369.1 Define Documentation

7.369.1.1 #define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)

7.369.1.2 #define ENOUGH_DISTS 592

7.369.1.3 #define ENOUGH_LENS 852

7.369.2 Enumeration Type Documentation

7.369.2.1 enum codetype

Enumerator:

CODES

LENS

DISTS

7.369.3 Function Documentation

7.369.3.1 int ZLIB_INTERNAL inflate_table OF ((codetype type, unsigned short FAR ∗lens, unsigned codes, code FAR ∗FAR ∗table, unsigned FAR ∗bits, unsigned short FAR ∗work))

7.370 src/main/decaf/internal/util/zip/trees.h File Reference

Variables

- **local const ct_data static_ltree** [**L_CODES**+2]
- **local const ct_data static_dtree** [**D_CODES**]
- **const uch ZLIB_INTERNAL _dist_code** [DIST_CODE_LEN]
- **const uch ZLIB_INTERNAL _length_code** [**MAX_MATCH**-**MIN_MATCH**+1]
- **local const** int **base_length** [**LENGTH_CODES**]
- **local const** int **base_dist** [**D_CODES**]

7.370.1 Variable Documentation

7.370.1.1 const uch ZLIB_INTERNAL _dist_code[DIST_CODE_LEN]

Initial value:

```
 {
 0,  1,  2,  3,  4,  4,  5,  5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  8,
 8,  8,  8,  8,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,  0,  0, 16, 17,
18, 18, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29
}
```

7.370.1.2 const uch ZLIB_INTERNAL _length_code[MAX_MATCH-MIN_MATCH+1]

Initial value:

```
 {
 0,  1,  2,  3,  4,  5,  6,  7,  8,  8,  9,  9, 10, 10, 11, 11, 12, 12, 12, 12,
13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16,
17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28
}
```

7.370.1.3 local const int base_dist[D_CODES]

Initial value:

```
 {
    0,     1,     2,     3,     4,     6,     8,    12,    16,    24,
   32,    48,    64,    96,   128,   192,   256,   384,   512,   768,
 1024,  1536,  2048,  3072,  4096,  6144,  8192, 12288, 16384, 24576
}
```

7.370.1.4 local const int base_length[LENGTH_CODES]

Initial value:

```
 {
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
}
```

7.370.1.5 local const ct_data static_dtree[D_CODES]

Initial value:

```
 {
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
}
```

7.370.1.6 local const ct_data static_ltree[L_CODES+2]

7.371 src/main/decaf/internal/util/zip/zconf.h File Reference

```
#include <decaf/util/Config.h>
```

Defines

- #define **const** /∗ note: need a more gentle solution here ∗/
- #define **MAX_MEM_LEVEL** 9
- #define **MAX_WBITS** 15 /∗ 32K LZ77 window ∗/
- #define **OF**(args) ()
- #define **ZEXTERN** extern
- #define **ZEXPORT**
- #define **ZEXPORTVA**
- #define **FAR**
- #define **SEEK_SET** 0 /∗ Seek from beginning of file. ∗/
- #define **SEEK_CUR** 1 /∗ Seek from current position. ∗/
- #define **SEEK_END** 2 /∗ Set file pointer to EOF plus "offset" ∗/
- #define **z_off_t** long
- #define **z_off64_t z_off_t**

Typedefs

- typedef unsigned char **Byte**
- typedef unsigned int **uInt**
- typedef unsigned long **uLong**
- typedef **Byte FAR Bytef**
- typedef char **FAR charf**
- typedef int **FAR intf**
- typedef **uInt FAR uIntf**
- typedef **uLong FAR uLongf**
- typedef **Byte const ∗ voidpc**
- typedef **Byte FAR ∗ voidpf**
- typedef **Byte ∗ voidp**

7.371.1 Define Documentation

7.371.1.1 #define const /∗ note: need a more gentle solution here ∗/

7.371.1.2 #define FAR

7.371.1.3 #define MAX_MEM_LEVEL 9

7.371.1.4 #define MAX_WBITS 15 /∗ 32K LZ77 window ∗/

7.371.1.5 #define OF(*args*) ()

7.371.1.6 #define SEEK_CUR 1 /∗ Seek from current position. ∗/

7.371.1.7 #define SEEK_END 2 /∗ Set file pointer to EOF plus "offset" ∗/

7.371.1.8 #define SEEK_SET 0 /∗ Seek from beginning of file. ∗/

7.371.1.9 #define z_off64_t z_off_t

7.371.1.10 #define z_off_t long

7.371.1.11 #define ZEXPORT

7.371.1.12 #define ZEXPORTVA

7.371.1.13 #define ZEXTERN extern

7.371.2 Typedef Documentation

7.371.2.1 typedef unsigned char Byte

7.371.2.2 typedef Byte FAR Bytef

7.371.2.3 typedef char FAR charf

7.371.2.4 typedef int FAR intf

7.371.2.5 typedef unsigned int uInt

7.371.2.6 typedef uInt FAR uIntf

7.371.2.7 typedef unsigned long uLong

7.371.2.8 typedef uLong FAR uLongf

7.371.2.9 typedef Byte* voidp

7.371.2.10 typedef Byte const* voidpc

7.371.2.11 typedef Byte FAR* voidpf

7.372 src/main/decaf/internal/util/zip/zlib.h File Reference

```
#include "zconf.h"
```

Data Structures

- struct **z_stream_s**
- struct **gz_header_s**
- struct **internal_state**

Defines

- #define **ZLIB_VERSION** "1.2.5"
- #define **ZLIB_VERNUM** 0x1250
- #define **ZLIB_VER_MAJOR** 1
- #define **ZLIB_VER_MINOR** 2
- #define **ZLIB_VER_REVISION** 5
- #define **ZLIB_VER_SUBREVISION** 0
- #define **Z_NO_FLUSH** 0
- #define **Z_PARTIAL_FLUSH** 1
- #define **Z_SYNC_FLUSH** 2
- #define **Z_FULL_FLUSH** 3
- #define **Z_FINISH** 4
- #define **Z_BLOCK** 5
- #define **Z_TREES** 6
- #define **Z_OK** 0
- #define **Z_STREAM_END** 1
- #define **Z_NEED_DICT** 2
- #define **Z_ERRNO** (-1)
- #define **Z_STREAM_ERROR** (-2)
- #define **Z_DATA_ERROR** (-3)
- #define **Z_MEM_ERROR** (-4)
- #define **Z_BUF_ERROR** (-5)
- #define **Z_VERSION_ERROR** (-6)
- #define **Z_NO_COMPRESSION** 0
- #define **Z_BEST_SPEED** 1
- #define **Z_BEST_COMPRESSION** 9
- #define **Z_DEFAULT_COMPRESSION** (-1)
- #define **Z_FILTERED** 1
- #define **Z_HUFFMAN_ONLY** 2
- #define **Z_RLE** 3
- #define **Z_FIXED** 4

- #define **Z_DEFAULT_STRATEGY** 0
- #define **Z_BINARY** 0
- #define **Z_TEXT** 1
- #define **Z_ASCII Z_TEXT** /* for compatibility with 1.2.2 and earlier */
- #define **Z_UNKNOWN** 2
- #define **Z_DEFLATED** 8
- #define **Z_NULL** 0 /* for initializing zalloc, zfree, opaque */
- #define **zlib_version** zlibVersion()
- #define **deflateInit**(strm, level) deflateInit_((strm), (level), **ZLIB_VERSION**, sizeof(**z_stream**))
- #define **inflateInit**(strm) inflateInit_((strm), **ZLIB_VERSION**, sizeof(**z_stream**))
- #define **deflateInit2**(strm, level, method, windowBits, memLevel, strategy)
- #define **inflateInit2**(strm, windowBits) inflateInit2_((strm), (windowBits), **ZLIB_VERSION**, sizeof(**z_stream**))
- #define **inflateBackInit**(strm, windowBits, window)

Typedefs

- typedef **voidpf** alloc_func **OF** ((**voidpf** opaque, **uInt** items, **uInt** size))
- typedef struct **z_stream_s z_stream**
- typedef **z_stream FAR** * **z_streamp**
- typedef struct **gz_header_s gz_header**
- typedef **gz_header FAR** * **gz_headerp**
- typedef **voidp gzFile**

Functions

- **ZEXTERN const** char *__ZEXPORT__
  zlibVersion **OF** ((void))
- **ZEXTERN** int **ZEXPORT** deflate **OF** ((**z_streamp** strm, int flush))
- **ZEXTERN** int **ZEXPORT** deflateEnd **OF** ((**z_streamp** strm))
- **ZEXTERN** int **ZEXPORT**
  deflateSetDictionary **OF** ((**z_streamp** strm, **const Bytef** *dictionary, uIntdictLength))
- **ZEXTERN** int **ZEXPORT** deflateCopy **OF** ((**z_streamp** dest, **z_streamp** source))
- **ZEXTERN** int **ZEXPORT** deflateParams **OF** ((**z_streamp** strm, int level, int strategy))
- **ZEXTERN** int **ZEXPORT** deflateTune **OF** ((**z_streamp** strm, int good_length, int max_lazy, int nice_length, int max_chain))
- **ZEXTERN uLong ZEXPORT** deflateBound **OF** ((**z_streamp** strm, **uLong** sourceLen))
- **ZEXTERN** int **ZEXPORT** deflatePrime **OF** ((**z_streamp** strm, int bits, int value))
- **ZEXTERN** int **ZEXPORT**
  deflateSetHeader **OF** ((**z_streamp** strm, **gz_headerp** head))
- **ZEXTERN** int **ZEXPORT** inflateReset2 **OF** ((**z_streamp** strm, int windowBits))
- **ZEXTERN** int **ZEXPORT** inflateBack **OF** ((**z_streamp** strm, in_func in, void **FAR** *in_desc, out_func out, void **FAR** *out_desc))
- **ZEXTERN** int **ZEXPORT** compress **OF** ((**Bytef** *dest, **uLongf** *destLen, **const Bytef** *source, **uLong** sourceLen))
- **ZEXTERN** int **ZEXPORT** compress2 **OF** ((**Bytef** *dest, **uLongf** *destLen, **const Bytef** *source, **uLong** sourceLen, int level))
- **ZEXTERN uLong ZEXPORT** compressBound **OF** ((**uLong** sourceLen))
- **ZEXTERN gzFile ZEXPORT** gzdopen **OF** ((int fd, **const** char *mode))
- **ZEXTERN** int **ZEXPORT** gzbuffer **OF** ((**gzFile** file, unsigned size))
- **ZEXTERN** int **ZEXPORT** gzsetparams **OF** ((**gzFile** file, int level, int strategy))
- **ZEXTERN** int **ZEXPORT** gzread **OF** ((**gzFile** file, **voidp** buf, unsigned len))
- **ZEXTERN** int **ZEXPORT** gzwrite **OF** ((**gzFile** file, **voidpc** buf, unsigned len))
- **ZEXTERN** int **ZEXPORTVA** gzprintf **OF** ((**gzFile** file, **const** char *format,...))
- **ZEXTERN** int **ZEXPORT** gzputs **OF** ((**gzFile** file, **const** char *s))

- **ZEXTERN** char ∗**ZEXPORT** gzgets **OF** ((**gzFile** file, char ∗buf, int len))
- **ZEXTERN** int **ZEXPORT** gzputc **OF** ((**gzFile** file, int c))
- **ZEXTERN** int **ZEXPORT** gzgetc **OF** ((**gzFile** file))
- **ZEXTERN** int **ZEXPORT** gzungetc **OF** ((int c, **gzFile** file))
- **ZEXTERN** int **ZEXPORT** gzflush **OF** ((**gzFile** file, int flush))
- **ZEXTERN const** char ∗**ZEXPORT** gzerror **OF** ((**gzFile** file, int ∗errnum))
- **ZEXTERN uLong ZEXPORT** adler32 **OF** ((**uLong** adler, **const Bytef** ∗buf, **uInt** len))
- **ZEXTERN uLong ZEXPORT** crc32 **OF** ((**uLong** crc, **const Bytef** ∗buf, **uInt** len))
- **ZEXTERN** int **ZEXPORT** deflateInit_ **OF** ((**z_streamp** strm, int level, **const** char ∗version, int stream_size))
- **ZEXTERN** int **ZEXPORT** inflateInit_ **OF** ((**z_streamp** strm, **const** char ∗version, int stream_size))
- **ZEXTERN** int **ZEXPORT** deflateInit2_ **OF** ((**z_streamp** strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, **const** char ∗version, int stream_size))
- **ZEXTERN** int **ZEXPORT** inflateInit2_ **OF** ((**z_streamp** strm, intwindowBits, **const** char ∗version, int stream_size))
- **ZEXTERN** int **ZEXPORT** inflateBackInit_ **OF** ((**z_streamp** strm, int windowBits, unsigned char **FAR** ∗window, **const** char ∗version, int stream_size))
- **ZEXTERN gzFile ZEXPORT** gzopen **OF** ((**const** char ∗, **const** char ∗))
- **ZEXTERN z_off_t ZEXPORT** gzseek **OF** ((**gzFile**, **z_off_t**, int))
- **ZEXTERN z_off_t ZEXPORT** gztell **OF** ((**gzFile**))
- **ZEXTERN uLong ZEXPORT** adler32_combine **OF** ((**uLong**, **uLong**, **z_off_t**))
- **ZEXTERN const** char ∗**ZEXPORT** zError **OF** ((int))
- **ZEXTERN** int **ZEXPORT** inflateSyncPoint **OF** ((**z_streamp**))
- **ZEXTERN** int **ZEXPORT** inflateUndermine **OF** ((**z_streamp**, int))

7.372.1 Define Documentation

7.372.1.1 #define deflateInit(*strm, level*) deflateInit_((strm), (level), ZLIB_VERSION, sizeof(z_stream))

7.372.1.2 #define deflateInit2(*strm, level, method, windowBits, memLevel, strategy*)

Value:

```
deflateInit2_((strm),(level),(method),(windowBits),(memLevel),\
                      (strategy),           ZLIB_VERSION, sizeof(z_stream))
```

7.372.1.3 #define inflateBackInit(*strm, windowBits, window*)

Value:

```
inflateBackInit_((strm), (windowBits), (window), \
                                          ZLIB_VERSION, sizeof(z_stream))
```

7.372.1.4 #define inflateInit(*strm*) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))

7.372.1.5 #define inflateInit2(*strm, windowBits*) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))

7.372.1.6 #define Z_ASCII Z_TEXT /∗ for compatibility with 1.2.2 and earlier ∗/

7.372.1.7 #define Z_BEST_COMPRESSION 9

7.372.1.8 #define Z_BEST_SPEED 1

7.372.1.9 #define Z_BINARY 0

7.372.1.10 #define Z_BLOCK 5

7.372.1.11 #define Z_BUF_ERROR (-5)

7.372.1.12 #define Z_DATA_ERROR (-3)

7.372.1.13 #define Z_DEFAULT_COMPRESSION (-1)

7.372.1.14 #define Z_DEFAULT_STRATEGY 0

7.372.1.15 #define Z_DEFLATED 8

7.372.1.16 #define Z_ERRNO (-1)

7.372.1.17 #define Z_FILTERED 1

7.372.1.18 #define Z_FINISH 4

7.372.1.19 #define Z_FIXED 4

7.372.1.20 #define Z_FULL_FLUSH 3

7.372.1.21 #define Z_HUFFMAN_ONLY 2

7.372.1.22 #define Z_MEM_ERROR (-4)

7.372.1.23 #define Z_NEED_DICT 2

7.372.1.24 #define Z_NO_COMPRESSION 0

7.372.1.25 #define Z_NO_FLUSH 0

7.372.1.26 #define Z_NULL 0 /* for initializing zalloc, zfree, opaque */

7.372.1.27 #define Z_OK 0

7.372.1.28 #define Z_PARTIAL_FLUSH 1

7.372.1.29 #define Z_RLE 3

7.372.1.30 #define Z_STREAM_END 1

7.372.1.31 #define Z_STREAM_ERROR (-2)

7.372.1.32 #define Z_SYNC_FLUSH 2

7.372.1.33 #define Z_TEXT 1

7.372.1.34 #define Z_TREES 6

7.372.1.35 #define Z_UNKNOWN 2

7.372.1.36 #define Z_VERSION_ERROR (-6)

7.372.1.37 #define ZLIB_VER_MAJOR 1

7.372.1.38 #define ZLIB_VER_MINOR 2

7.372.1.39 #define ZLIB_VER_REVISION 5

7.372.1.40 #define ZLIB_VER_SUBREVISION 0

7.372.1.41 #define ZLIB_VERNUM 0x1250

7.372.1.42 #define ZLIB_VERSION "1.2.5"

7.372.1.43 #define zlib_version zlibVersion()

7.372.2 Typedef Documentation

7.372.2.1 typedef struct gz_header_s gz_header

7.372.2.2 typedef gz_header FAR* gz_headerp

7.372.2.3 typedef voidp gzFile

7.372.2.4 ZEXTERN uLong ZEXPORT crc32_combine64 OF ((voidpf opaque, uInt items, uInt size))

7.372.2.5 typedef struct z_stream_s z_stream

7.372.2.6 typedef z_stream FAR* z_streamp

7.372.3 Function Documentation

7.372.3.1 ZEXTERN const char* ZEXPORT zlibVersion OF ((void))

7.372.3.2 ZEXTERN int ZEXPORT deflate OF ((z_streamp strm, int flush))

7.372.3.3 ZEXTERN int ZEXPORT deflateEnd OF ((z_streamp strm))

7.372.3.4 ZEXTERN int ZEXPORT deflateSetDictionary OF ((z_streamp strm, const Bytef *dictionary, uInt dictLength))

7.372.3.5 ZEXTERN int ZEXPORT deflateCopy OF ((z_streamp dest, z_streamp source))

7.372.3.6 ZEXTERN int ZEXPORT deflateParams OF ((z_streamp strm, int level, int strategy))

7.372.3.7 ZEXTERN int ZEXPORT deflateTune OF ((z_streamp strm, int good_length, int max_lazy, int nice_length, int max_chain))

7.372.3.8 ZEXTERN uLong ZEXPORT deflateBound OF ((z_streamp strm, uLong sourceLen))

7.372.3.9 ZEXTERN int ZEXPORT deflatePrime OF ((z_streamp strm, int bits, int value))

7.372.3.10 ZEXTERN int ZEXPORT deflateSetHeader OF ((z_streamp strm, gz_headerp head))

7.372.3.11 ZEXTERN int ZEXPORT inflateReset2 OF ((z_streamp strm, int windowBits))

7.372.3.12 ZEXTERN int ZEXPORT inflateBack OF ((z_streamp strm, in_func in, void FAR *in_desc, out_func out, void FAR *out_desc))

7.372.3.13 ZEXTERN int ZEXPORT compress OF ((Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen))

7.372.3.14 ZEXTERN int ZEXPORT compress2 OF ((Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen, int level))

7.372.3.15 ZEXTERN uLong ZEXPORT compressBound OF ((uLong sourceLen))

7.372.3.16 ZEXTERN gzFile ZEXPORT gzdopen OF ((int fd, const char *mode))

7.372.3.17 ZEXTERN int ZEXPORT gzbuffer OF ((gzFile file, unsigned size))

7.372.3.18 ZEXTERN int ZEXPORT gzsetparams OF ((gzFile file, int level, int strategy))

7.372.3.19 ZEXTERN int ZEXPORT gzread OF ((gzFile file, voidp buf, unsigned len))

7.372.3.20 ZEXTERN int ZEXPORT gzwrite OF ((gzFile file, voidpc buf, unsigned len))

7.372.3.21 ZEXTERN int ZEXPORTVA gzprintf OF ((gzFile file, const char *format,...))

7.372.3.22 ZEXTERN int ZEXPORT gzputs OF ((gzFile file, const char *s))

7.372.3.23 ZEXTERN char* ZEXPORT gzgets OF ((gzFile file, char *buf, int len))

7.372.3.24 ZEXTERN int ZEXPORT gzputc OF ((gzFile file, int c))

7.372.3.25 ZEXTERN int ZEXPORT gzgetc OF ((gzFile file))

7.372.3.26 ZEXTERN int ZEXPORT gzungetc OF ((int c, gzFile file))

7.372.3.27 ZEXTERN int ZEXPORT gzflush OF ((gzFile file, int flush))

7.372.3.28 ZEXTERN const char* ZEXPORT gzerror OF ((gzFile file, int *errnum))

7.372.3.29 ZEXTERN uLong ZEXPORT adler32 OF ((uLong adler, const Bytef *buf, uInt len))

7.372.3.30 ZEXTERN uLong ZEXPORT crc32 OF ((uLong crc, const Bytef *buf, uInt len))

7.372.3.31 ZEXTERN int ZEXPORT deflateInit_ OF ((z_streamp strm, int level, const char *version, int stream_size))

7.372.3.32 ZEXTERN int ZEXPORT inflateInit_ OF ((z_streamp strm, const char *version, int stream_size))

7.372.3.33 ZEXTERN int ZEXPORT deflateInit2_ OF ((z_streamp strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, const char *version, int stream_size))

7.372.3.34 ZEXTERN int ZEXPORT inflateInit2_ OF ((z_streamp strm, intwindowBits, const char *version, int stream_size))

7.372.3.35 ZEXTERN int ZEXPORT inflateBackInit_ OF ((z_streamp strm, int windowBits, unsigned char FAR *window, const char *version, int stream_size))

7.372.3.36 ZEXTERN gzFile ZEXPORT gzopen OF ((const char *, const char *))

7.372.3.37 ZEXTERN z_off_t ZEXPORT gzseek OF ((gzFile, z_off_t, int))

7.372.3.38 ZEXTERN z_off_t ZEXPORT gztell OF ((gzFile))

7.372.3.39 ZEXTERN uLong ZEXPORT adler32_combine OF ((uLong, uLong, z_off_t))

7.372.3.40 ZEXTERN const char* ZEXPORT zError OF ((int))

7.372.3.41 ZEXTERN int ZEXPORT inflateSyncPoint OF ((z_streamp))

7.372.3.42 ZEXTERN int ZEXPORT inflateUndermine OF ((z_streamp, int))

7.373 src/main/decaf/internal/util/zip/zutil.h File Reference

```
#include "zlib.h"
```

Defines

- #define **ZLIB_INTERNAL**
- #define **local** static
- #define **ERR_MSG**(err) **z_errmsg**[**Z_NEED_DICT**-(err)]
- #define **ERR_RETURN**(strm, err) return (strm->msg = (char*)**ERR_MSG**(err), (err))
- #define **DEF_WBITS MAX_WBITS**
- #define **DEF_MEM_LEVEL** 8
- #define **STORED_BLOCK** 0
- #define **STATIC_TREES** 1
- #define **DYN_TREES** 2
- #define **MIN_MATCH** 3
- #define **MAX_MATCH** 258
- #define **PRESET_DICT** 0x20 /* preset dictionary flag in zlib header */
- #define **OS_CODE** 0x03 /* assume Unix */
- #define **F_OPEN**(name, mode) fopen((name), (mode))
- #define **Assert**(cond, msg)
- #define **Trace**(x)
- #define **Tracev**(x)
- #define **Tracevv**(x)
- #define **Tracec**(c, x)
- #define **Tracecv**(c, x)
- #define **ZALLOC**(strm, items, size) (*((strm)->zalloc))((strm)->opaque, (items), (size))
- #define **ZFREE**(strm, addr) (*((strm)->zfree))((strm)->opaque, (**voidpf**)(addr))
- #define **TRY_FREE**(s, p) {if (p) **ZFREE**(s, p);}

Typedefs

- typedef unsigned char **uch**
- typedef **uch FAR uchf**
- typedef unsigned short **ush**
- typedef **ush FAR ushf**
- typedef unsigned long **ulg**

Functions

- **ZEXTERN uLong ZEXPORT**
  adler32_combine64 **OF** ((**uLong**, **uLong**, **z_off_t**))
- void **ZLIB_INTERNAL** zmemcpy **OF** ((**Bytef** *dest, **const Bytef** *source, **uInt** len))
- int **ZLIB_INTERNAL** zmemcmp **OF** ((**const Bytef** *s1, **const Bytef** *s2, **uInt** len))
- void **ZLIB_INTERNAL** zmemzero **OF** ((**Bytef** *dest, **uInt** len))
- **voidpf ZLIB_INTERNAL** zcalloc **OF** ((**voidpf** opaque, unsigned items, unsigned size))
- void **ZLIB_INTERNAL** zcfree **OF** ((**voidpf** opaque, **voidpf** ptr))

Variables

- **const** char ***const z_errmsg** [10]

7.373.1 Define Documentation

7.373.1.1 #define **Assert**(*cond, msg*)

7.373.1.2 #define **DEF_MEM_LEVEL** 8

7.373.1.3 #define **DEF_WBITS MAX_WBITS**

7.373.1.4 #define **DYN_TREES** 2

7.373.1.5 #define **ERR_MSG**(*err*) **z_errmsg**[Z_NEED_DICT-(err)]

7.373.1.6 #define **ERR_RETURN**(*strm, err*) return (strm->msg = (char*)**ERR_MSG**(err), (err))

7.373.1.7 #define **F_OPEN**(*name, mode*) fopen((name), (mode))

7.373.1.8 #define **local** static

7.373.1.9 #define **MAX_MATCH** 258

7.373.1.10 #define **MIN_MATCH** 3

7.373.1.11 #define **OS_CODE** 0x03 /* assume Unix */

7.373.1.12 #define **PRESET_DICT** 0x20 /* preset dictionary flag in zlib header */

7.373.1.13 #define **STATIC_TREES** 1

7.373.1.14 #define **STORED_BLOCK** 0

7.373.1.15 #define **Trace**(*x*)

7.373.1.16 #define **Tracec**(*c, x*)

7.373.1.17 #define **Tracecv**(*c, x*)

7.373.1.18 #define **Tracev**(*x*)

7.373.1.19 #define **Tracevv**(*x*)

7.373.1.20 #define TRY_FREE(*s, p*) {if (p) ZFREE(s, p);}

7.373.1.21 #define ZALLOC(*strm, items, size*) (*((strm)->zalloc))((strm)->opaque, (items), (size))

7.373.1.22 #define ZFREE(*strm, addr*) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))

7.373.1.23 #define ZLIB_INTERNAL

7.373.2 Typedef Documentation

7.373.2.1 typedef unsigned char uch

7.373.2.2 typedef uch FAR uchf

7.373.2.3 typedef unsigned long ulg

7.373.2.4 typedef unsigned short ush

7.373.2.5 typedef ush FAR ushf

7.373.3 Function Documentation

7.373.3.1 ZEXTERN uLong ZEXPORT adler32_combine64 OF ((uLong, uLong, z_off_t))

7.373.3.2 void ZLIB_INTERNAL zmemcpy OF ((Bytef *dest, const Bytef *source, uInt len))

7.373.3.3 int ZLIB_INTERNAL zmemcmp OF ((const Bytef *s1, const Bytef *s2, uInt len))

7.373.3.4 void ZLIB_INTERNAL zmemzero OF ((Bytef *dest, uInt len))

7.373.3.5 voidpf ZLIB_INTERNAL zcalloc OF ((voidpf opaque, unsigned items, unsigned size))

7.373.3.6 void ZLIB_INTERNAL zcfree OF ((voidpf opaque, voidpf ptr))

7.373.4 Variable Documentation

7.373.4.1 const char* const z_errmsg[10]

7.374 src/main/decaf/io/BlockingByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.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**

7.375 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#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**

7.376 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**

7.377 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::ByteArrayInputStream**

  *A **ByteArrayInputStream** (p. 527) contains an internal buffer that contains bytes that may be read from the stream.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.378 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <utility>
```

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**

7.379 src/main/decaf/io/DataInput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#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::DataInput**

  *The **DataInput** (p. 842) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.380 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**

7.381 src/main/decaf/io/DataOutput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataOutput**

  *The **DataOutput** (p. 856) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.382 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**

7.383 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**

7.384 src/main/decaf/io/FileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::FileDescriptor**

  This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.385 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::FilterInputStream**

  *A **FilterInputStream** (p. 1032) 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**

7.386 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.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**

7.387 src/main/decaf/io/Flushable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::Flushable**

  *A **Flushable** (p. 1067) is a destination of data that can be flushed.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.388 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/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::InputStream**

  A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.389 src/main/decaf/io/InputStreamReader.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/Reader.h>
```

Data Structures

- class **decaf::io::InputStreamReader**

  *An **InputStreamReader** (p. 1142) is a bridge from byte streams to character streams.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.390 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**

7.391 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**

7.392 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::OutputStream**

  Base interface for any class that wants to represent an output stream of bytes.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.393 src/main/decaf/io/OutputStreamWriter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/Writer.h>
```

Data Structures

- class **decaf::io::OutputStreamWriter**

  A class for turning a character stream into a byte stream.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.394 src/main/decaf/io/PushbackInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/FilterInputStream.h>
```

Data Structures

- class **decaf::io::PushbackInputStream**

  *A **PushbackInputStream** (p. 1716) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.395 src/main/decaf/io/Reader.h File Reference

```
#include <string>
```

```
#include <decaf/lang/Readable.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.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**

7.396 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**

7.397 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**

7.398 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <vector>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/lang/Appendable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.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**

7.399 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Appendable**

  An object to which char sequences and values can be appended.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.400 src/main/decaf/lang/ArrayPointer.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/System.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <decaf/util/Arrays.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

Data Structures

- class **decaf::lang::ArrayPointer< T, REFCOUNTER >**

  *Decaf's implementation of a Smart **Pointer** (p. 1614) that is a template on a Type and is **Thread** (p. 2094) Safe if the default Reference Counter is used.*
- struct **decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayData**
- class **decaf::lang::ArrayPointerComparator< 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 value of the actual pointer to the array being contained in this **ArrayPointer** (p. 358).*
- struct **std::less< decaf::lang::ArrayPointer< 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** ArrayPointer< T, R > &left, **const** U *right)
- template<typename T , typename R , typename U >
  bool **decaf::lang::operator==** (**const** U *left, **const** ArrayPointer< T, R > &right)
- template<typename T , typename R , typename U >
  bool **decaf::lang::operator!=** (**const** ArrayPointer< T, R > &left, **const** U *right)
- template<typename T , typename R , typename U >
  bool **decaf::lang::operator!=** (**const** U *left, **const** ArrayPointer< T, R > &right)

7.401 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**

7.402 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**

7.403 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**

7.404 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. 623) 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**

7.405 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**

7.406 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**

7.407 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**

7.408 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**

7.409 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**

7.410 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**

7.411 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**

7.412 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**

7.413 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**

7.414 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**

7.415 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**

7.416 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**

7.417 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**

7.418 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**

7.419 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**

7.420 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. *1207*) *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**

7.421 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**

7.422 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Math**

  The class ***Math*** *(p. 1396) 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**

7.423 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Number**

  The abstract class ***Number*** *(p. 1543) is the superclass of classes* ***Byte*** *(p. 476),* ***Double*** *(p. 956),* ***Float*** *(p. 1040),* ***Integer*** *(p. 1161),* ***Long*** *(p. 1338), and* ***Short*** *(p. 1858).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.424 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/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
#include <functional>
```

Data Structures

- 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. 1614) that is a template on a Type and is **Thread** (p. 2094) 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. 1614) 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)

7.425 src/main/decaf/lang/Readable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::lang::Readable**

  *A **Readable** (p. 1732) is a source of characters.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**
- namespace **decaf::lang**

7.426 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**

7.427 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**

7.428 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**

7.429 src/main/decaf/lang/String.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::lang::String**

  *The **String** (p. 2031) class represents an immutable sequence of chars.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.430 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/internal/AprPool.h>
#include <string>
```

Data Structures

- class **decaf::lang::System**

  *The **System** (p. 2065) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.431 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/util/Config.h>
```

Data Structures

- class **decaf::lang::Thread**

  *A **Thread** (p. 2094) is a concurrent unit of execution.*
- class **decaf::lang::Thread::UncaughtExceptionHandler**

  *Interface for handlers invoked when a **Thread** (p. 2094) abruptly terminates due to an uncaught 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**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

7.432 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**

7.433 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**

7.434 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**

7.435 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**

7.436 src/main/decaf/net/DatagramPacket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/Inet4Address.h>
#include <decaf/net/SocketAddress.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::net::DatagramPacket**

  Class that represents a single datagram packet.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.437 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**

7.438 src/main/decaf/net/Inet4Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

Data Structures

- class **decaf::net::Inet4Address**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.439 src/main/decaf/net/Inet6Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

Data Structures

- class **decaf::net::Inet6Address**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.440 src/main/decaf/net/InetAddress.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/ArrayPointer.h>
```

Data Structures

- class **decaf::net::InetAddress**

  Represents an IP address.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.441 src/main/decaf/net/InetSocketAddress.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketAddress.h>
#include <string>
```

Data Structures

- class **decaf::net::InetSocketAddress**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.442 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**

7.443 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**

7.444 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**

7.445 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**

7.446 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
#include <string>
```

Data Structures

- class **decaf::net::ServerSocket**

  This class implements server sockets.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.447 src/main/decaf/net/ServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

Data Structures

- class **decaf::net::ServerSocketFactory**

  Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.448 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImplFactory.h>
#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 <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.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**

7.449 src/main/decaf/net/SocketAddress.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketAddress**

  *Base class for protocol specific **Socket** (p. 1900) addresses.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.450 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**

7.451 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**

7.452 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/net/UnknownHostException.h>
```

Data Structures

- class **decaf::net::SocketFactory**

  *The **SocketFactory** (p. 1916) is used to create **Socket** (p. 1900) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.453 src/main/decaf/net/SocketImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/FileDescriptor.h>
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/net/SocketOptions.h>
#include <string>
```

Data Structures

- class **decaf::net::SocketImpl**

  *Acts as a base class for all physical **Socket** (p. 1900) implementations.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.454 src/main/decaf/net/SocketImplFactory.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketImplFactory**

  Factory class interface for a Factory that creates ScoketImpl objects.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.455 src/main/decaf/net/SocketOptions.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketOptions**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.456 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**

7.457 src/main/decaf/net/ssl/SSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

Data Structures

- class **decaf::net::ssl::SSLContext**

  *Represents on implementation of the Secure **Socket** (p. 1900) Layer for streaming based sockets.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.458 src/main/decaf/net/ssl/SSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandom.h>
```

Data Structures

- class **decaf::net::ssl::SSLContextSpi**

  *Defines the interface that should be provided by an **SSLContext** (p. 1934) provider.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.459 src/main/decaf/net/ssl/SSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

Data Structures

- class **decaf::net::ssl::SSLParameters**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.460 src/main/decaf/net/ssl/SSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocket.h>
```

Data Structures

- class **decaf::net::ssl::SSLServerSocket**

  Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.461 src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
#include <vector>
#include <string>
```

Data Structures

- class **decaf::net::ssl::SSLServerSocketFactory**

  Factory class interface that provides methods to create SSL Server Sockets.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.462 src/main/decaf/net/ssl/SSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/Socket.h>
#include <decaf/net/ssl/SSLParameters.h>
```

Data Structures

- class **decaf::net::ssl::SSLSocket**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.463 src/main/decaf/net/ssl/SSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
#include <vector>
#include <string>
```

Data Structures

- class **decaf::net::ssl::SSLSocketFactory**

  *Factory class interface for a **SocketFactory** (p. 1916) that can create **SSLSocket** (p. 1947) objects.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**
- namespace **decaf::net::ssl**

7.464 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**

7.465 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**

7.466 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. *2191) 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**

7.467 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**

7.468 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URL**

  *Class **URL** (p. 2223) 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**

7.469 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**

7.470 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**

7.471 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**

7.472 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**

7.473 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**

7.474 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>
```

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**

7.475 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**

7.476 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**

7.477 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**

7.478 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**

7.479 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**

7.480 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**

7.481 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**

7.482 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**

7.483 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**

7.484 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**

7.485 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**

7.486 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**

7.487 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**

7.488 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**

7.489 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**

7.490 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**

7.491 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**

7.492 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**

7.493 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. 1239) 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**

7.494 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**

7.495 src/main/decaf/security/KeyManagementException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/KeyException.h>
```

Data Structures

- class **decaf::security::KeyManagementException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.496 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**

7.497 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**

7.498 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**

7.499 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**

7.500 src/main/decaf/security/SecureRandom.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Random.h>
#include <decaf/security/SecureRandomSpi.h>
#include <memory>
```

Data Structures

- class **decaf::security::SecureRandom**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.501 src/main/decaf/security/SecureRandomSpi.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::SecureRandomSpi**

  Interface class used by Security Service Providers to implement a source of secure random bytes.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.502 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**

7.503 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. 660) 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**

7.504 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/ConcurrentModificationException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
#include <decaf/util/AbstractCollection.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractList< E >**

  *This class provides a skeletal implementation of the **List** (p. 1286) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*
- class **decaf::util::AbstractList< E >::SimpleListIterator**
- class **decaf::util::AbstractList< E >::ConstSimpleListIterator**

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**

7.505 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. 1371) 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**

7.506 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. 1723) operations.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.507 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. 1286) 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**

7.508 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. 1857) 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**

7.509 src/main/decaf/util/ArrayList.h File Reference

```
#include <memory>
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/System.h>
#include <decaf/lang/Integer.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
#include <decaf/util/AbstractList.h>
```

Data Structures

- class **decaf::util::ArrayList**< **E** >

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.510 src/main/decaf/util/Arrays.h File Reference

```
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::util::Arrays**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.511 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**

7.512 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
```

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**

7.513 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. 1250) **Comparator** (p. 689) 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**

7.514 src/main/decaf/util/concurrent/AbstractExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Executor.h>
#include <decaf/util/concurrent/ExecutorService.h>
```

Data Structures

- class **decaf::util::concurrent::AbstractExecutorService**

  *Provides a default implementation for the methods of the **ExecutorService** (p. 1008) 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**

7.515 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**

7.516 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**

7.517 src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference

```
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicRefCounter**

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**

7.518 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**

7.519 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>
```

Data Structures

- class **decaf::util::concurrent::BlockingQueue< E >**

  *A **decaf::util::Queue** (p. 1723) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.520 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**

7.521 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**

7.522 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**

7.523 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)

7.523.1 Define Documentation

7.523.1.1 #define synchronized(*W*)

Value:

```
if(false){}                                                    \
        else                                                        \
        for( decaf::util::concurrent::Lock lock_W(W);               \
             lock_W.isLocked(); lock_W.unlock() )
```

7.523.1.2 #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 compiles 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. 2533) { // Do something that needs synchronizing. } }

7.524 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/IllegalStateException.h>
#include <decaf/util/NoSuchElementException.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >**

  *Interface for a **Map** (p. 1371) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1371) 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**

7.525 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/util/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. 1371) 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**

7.526 src/main/decaf/util/concurrent/CopyOnWriteArrayList.h File Reference

```
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/ArrayPointer.h>
#include <decaf/lang/System.h>
#include <decaf/util/List.h>
```

Data Structures

- class **decaf::util::concurrent::CopyOnWriteArrayList< E >**
- class **decaf::util::concurrent::CopyOnWriteArrayList< E >::ArrayListIterator**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.527 src/main/decaf/util/concurrent/CopyOnWriteArraySet.h File Reference

```
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/ArrayPointer.h>
#include <decaf/util/concurrent/CopyOnWriteArrayList.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Set.h>
#include <decaf/util/Arrays.h>
#include <decaf/util/AbstractSet.h>
```

Data Structures

- class **decaf::util::concurrent::CopyOnWriteArraySet< E >**

  *Since the **CopyOnWriteArraySet** (p. 813) and the **CopyOnWriteArrayList** (p. 798) share much of the same operational semantics this class uses the **CopyOnWriteArrayList** (p. 798) for all its underlying operations.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.528 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.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**

7.529 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**

7.530 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**

7.531 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. 1792) 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**

7.532 src/main/decaf/util/concurrent/Executors.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/ExecutorService.h>
#include <decaf/util/concurrent/ThreadFactory.h>
```

Data Structures

- class **decaf::util::concurrent::Executors**

  *Implements a set of utilities for use with **Executors** (p. 1005), **ExecutorService** (p. 1008), **ThreadFactory** (p. 2102), and **Callable** (p. 578) types, as well as providing factory methods for instance of these types configured for the most common use cases.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.533 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/ArrayList.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. 1004) that provides methods to manage termination and methods that can produce a **Future** (p. 1075) 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**

7.534 src/main/decaf/util/concurrent/Future.h File Reference

```
#include <decaf/util/concurrent/TimeUnit.h>
```

Data Structures

- class **decaf::util::concurrent::Future**< **V** >

  *A **Future** (p. 1075) 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**

7.535 src/main/decaf/util/concurrent/LinkedBlockingQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Lock.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/Iterator.h>
#include <decaf/lang/Integer.h>
#include <decaf/lang/Math.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::util::concurrent::LinkedBlockingQueue< E >**

  *A **BlockingQueue** (p. 417) derivative that allows for a bound to be placed on the number of elements that can be enqueued at any one time.*
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::QueueNode< U >**
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::TotalLock**
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::LinkedIterator**
- class **decaf::util::concurrent::LinkedBlockingQueue< E >::ConstLinkedIterator**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.536 src/main/decaf/util/concurrent/Lock.h File Reference

```
#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**

7.537 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. *1305*) *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**

7.538 src/main/decaf/util/concurrent/locks/AbstractOwnableSynchronizer.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::locks::AbstractOwnableSynchronizer**

  Base class for locks that provide the notion of Ownership, the types of locks that are implemented using this base class would be owned by one specific Thread at any given time.

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**

7.539 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. *715) factors out the* ***Mutex*** *(*p. *1519) 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. *1305) 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**

7.540 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**

7.541 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**

  A ***ReadWriteLock*** *(*p. *1741) 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**

7.542 src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/locks/Lock.h>
```

Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**

  *A reentrant mutual exclusion **Lock*** (p. *1305*) *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**

7.543 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. *1519*) *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**

7.544 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**

7.545 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. 2104).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.546 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**

7.547 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**

7.548 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 **blocking queue** (p. 417) 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**

7.549 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
```

Data Structures

- class **decaf::util::concurrent::ThreadFactory**

  *public interface **ThreadFactory*** (p. *2102*)

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.550 src/main/decaf/util/concurrent/ThreadPoolExecutor.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Throwable.h>
#include <decaf/util/concurrent/ThreadFactory.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/util/concurrent/AbstractExecutorService.h>
#include <decaf/util/concurrent/RejectedExecutionHandler.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
#include <decaf/util/LinkedList.h>
#include <decaf/util/ArrayList.h>
#include <decaf/util/Config.h>
#include <vector>
```

Data Structures

- class **decaf::util::concurrent::ThreadPoolExecutor**

  Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.
- class **decaf::util::concurrent::ThreadPoolExecutor::AbortPolicy**

  *Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute*** (p. *2110*) *this class always throws a **RejectedExecutionException*** (p. *1755*).
- class **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy**

  *Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute*** (p. *2110*) *this class will attempt to run the task in the Thread that called the execute method unless the executor is shutdown in which case the task is not run and is destroyed.*
- class **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardPolicy**

  *Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute*** (p. *2110*) *this class always destroys the rejected task and returns quietly.*
- class **decaf::util::concurrent::ThreadPoolExecutor::CallerRunsPolicy::DiscardOldestPolicy**

  *Handler policy for tasks that are rejected upon a call to **ThreadPoolExecutor::execute*** (p. *2110*) *this class always destroys the oldest unexecuted task in the **Queue*** (p. *1723*) *and then attempts to execute the rejected task using the passed in executor.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.551 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**

7.552 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. 2134) 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**

7.553 src/main/decaf/util/ConcurrentModificationException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/RuntimeException.h>
```

Data Structures

- class **decaf::util::ConcurrentModificationException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.554 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**

7.555 src/main/decaf/util/Deque.h File Reference

```
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/util/Config.h>
#include <decaf/util/Queue.h>
```

Data Structures

- class **decaf::util::Deque**< **E** >

  *Defines a 'Double ended **Queue** (*p. *1723)' interface that allows for insertion and removal of elements from both ends.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.556 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::util::Iterator**< **E** >

  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**

7.557 src/main/decaf/util/LinkedList.h File Reference

```
#include <list>
#include <memory>
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/System.h>
#include <decaf/lang/Integer.h>
#include <decaf/util/Config.h>
#include <decaf/util/Deque.h>
#include <decaf/util/ArrayList.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/AbstractSequentialList.h>
```

Data Structures

- class **decaf::util::LinkedList**< **E** >

  *A complete implementation of the **List** (p. 1286) interface using a doubly linked list data structure.*
- class **decaf::util::LinkedList**< **E** >**::ListNode**< **U** >
- class **decaf::util::LinkedList**< **E** >**::LinkedListIterator**
- class **decaf::util::LinkedList**< **E** >**::ConstLinkedListIterator**
- class **decaf::util::LinkedList**< **E** >**::ReverseIterator**
- class **decaf::util::LinkedList**< **E** >**::ConstReverseIterator**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.558 src/main/decaf/util/List.h File Reference

```
#include <decaf/util/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/Collection.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**

7.559 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**

7.560 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/StreamHandler.h>
#include <decaf/util/logging/SimpleFormatter.h>
#include <decaf/io/IOException.h>
#include <decaf/internal/io/StandardErrorOutputStream.h>
```

Data Structures

- class **decaf::util::logging::ConsoleHandler**

  *This **Handler** (p. 1085) publishes log records to System.err.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.561 src/main/decaf/util/logging/ErrorManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::ErrorManager**

  ***ErrorManager** (p. 988) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1085) during Logging.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.562 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. 1031) 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**

7.563 src/main/decaf/util/logging/Formatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Handler.h>
```

Data Structures

- class **decaf::util::logging::Formatter**

  *A **Formatter** (p. 1074) 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**

7.564 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/Level.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::Handler**

  *A **Handler** (p. 1085) object takes log messages from a **Logger** (p. 1312) 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**

7.565 src/main/decaf/util/logging/Level.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::logging::Level**

  *The **Level** (p. 1253) class defines a set of standard logging levels that can be used to control logging output.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.566 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/LogManager.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <list>
#include <string>
#include <stdarg.h>
```

Data Structures

- class **decaf::util::logging::Logger**

  *A **Logger** (p. 1312) object is used to log messages for a specific system or application component.*

Namespaces

- namespace **decaf**

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

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.567 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::Levels** {
  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.

7.568 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);

7.568.1 Define Documentation

7.568.1.1 #define LOGDECAF_DEBUG(*logger, message*) logger.debug(__FILE__, __LINE__, message);

7.568.1.2 #define LOGDECAF_DEBUG_1(*logger, message, value*)

Value:

```
;                       \
   {                                                                \
      std::ostringstream ostream;                                   \
      ostream << message << value;                                  \
      logger.debug(__FILE__, __LINE__, ostream.str());              \
   }
```

7.568.1.3 #define **LOGDECAF_DECLARE(** *loggerName* **)** static **decaf::util::logging::SimpleLogger** loggerName;

7.568.1.4 #define **LOGDECAF_DECLARE_LOCAL(** *loggerName* **)** **decaf::util::logging::Logger** loggerName;

7.568.1.5 #define **LOGDECAF_ERROR(** *logger, message* **)** logger.error(__FILE__, __LINE__, message);

7.568.1.6 #define **LOGDECAF_FATAL(** *logger, message* **)** logger.fatal(__FILE__, __LINE__, message);

7.568.1.7 #define **LOGDECAF_INFO(** *logger, message* **)** logger.info(__FILE__, __LINE__, message);

7.568.1.8 #define **LOGDECAF_INITIALIZE(** *loggerName, className, loggerFamily* **)** **decaf::util::logging::SimpleLogger** className::loggerName(loggerFamily);

7.568.1.9 #define **LOGDECAF_WARN(** *logger, message* **)** logger.warn(__FILE__, __LINE__, message);

7.569 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**

7.570 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::logging::LogManager**

*There is a single global **LogManager** (p. 1327) 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::lang**
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.571 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Level.h>
#include <decaf/util/Config.h>
#include <memory>
#include <string>
```

Data Structures

- class **decaf::util::logging::LogRecord**

  ***LogRecord** (p. 1333) objects are used to pass logging requests between the logging framework and individual log Handlers.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.572 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**

7.573 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**

7.574 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. *1705*).

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.575 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::SimpleFormatter**

  *Print a brief summary of the **LogRecord*** (p. *1333*) *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**

7.576 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**

7.577 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.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**

  *Stream based logging **Handler*** (p. *1085*).

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.578 src/main/decaf/util/logging/XMLFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::XMLFormatter**

  *Format a **LogRecord** (p. 1333) into a standard XML 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**

7.579 src/main/decaf/util/Map.h File Reference

```
#include <functional>
#include <vector>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/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. 1371) 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**

7.580 src/main/decaf/util/NoSuchElementException.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
```

Data Structures

- class **decaf::util::NoSuchElementException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.581 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**

7.582 src/main/decaf/util/Properties.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/Pointer.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::io**
- namespace **decaf::util**

7.583 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. *1728) 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**

7.584 src/main/decaf/util/Set.h File Reference

```
#include <decaf/util/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**

7.585 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
#include <decaf/util/AbstractList.h>
```

Data Structures

- class **decaf::util::StlList< E >**

  List (p. *1286*) *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**

7.586 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/util/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. *1371*) *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**

7.587 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. 1723) 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**

7.588 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/util/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. 1857) 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**

7.589 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/util/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::util::StringTokenizer**

  Class that allows for parsing of string based on Tokens.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.590 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**

7.591 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. 2122).*

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**

7.592 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. 2230)).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.593 src/main/decaf/util/zip/Adler32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

Data Structures

- class **decaf::util::zip::Adler32**

  *Clas that can be used to compute an Adler-32 **Checksum** (p. 628) for a data stream.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.594 src/main/decaf/util/zip/CheckedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterInputStream.h>
```

Data Structures

- class **decaf::util::zip::CheckedInputStream**

  An implementation of a FilterInputStream that will maintain a ***Checksum*** *(*p. *628) of the bytes read, the* ***Checksum*** *(*p. *628) can then be used to verify the integrity of the input stream.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.595 src/main/decaf/util/zip/CheckedOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterOutputStream.h>
```

Data Structures

- class **decaf::util::zip::CheckedOutputStream**

  An implementation of a FilterOutputStream that will maintain a ***Checksum*** *(*p. *628) of the bytes written, the* ***Checksum*** *(*p. *628) can then be used to verify the integrity of the output stream.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.596 src/main/decaf/util/zip/Checksum.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::Checksum**

  *An interface used to represent **Checksum** (p. 628) values in the Zip package.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.597 src/main/decaf/util/zip/CRC32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

Data Structures

- class **decaf::util::zip::CRC32**

  Class that can be used to compute a CRC-32 checksum for a data stream.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.598 src/main/decaf/util/zip/DataFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::zip::DataFormatException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.599 src/main/decaf/util/zip/Deflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::Deflater**

  This class compresses data using the DEFLATE algorithm (see `specification`*).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.600 src/main/decaf/util/zip/DeflaterOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Deflater.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::DeflaterOutputStream**

  Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.601 src/main/decaf/util/zip/Inflater.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/zip/DataFormatException.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::Inflater**

  This class uncompresses data that was compressed using the DEFLATE algorithm (see `specification`*).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.602 src/main/decaf/util/zip/InflaterInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Inflater.h>
#include <vector>
```

Data Structures

- class **decaf::util::zip::InflaterInputStream**

  A FilterInputStream that decompresses data read from the wrapped InputStream instance.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

7.603 src/main/decaf/util/zip/ZipException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::util::zip::ZipException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::zip**

Index