log4c

1.2.4

Generated by Doxygen 1.8.11

Contents

1 Log4c : Logging for C Library **2**

1.1 Summary . . . 2

1.2 Requirements . . . 2

1.2.1 Platforms . . . 2

1.2.2 Software . . . 3

1.3 Installation . . . 3

1.3.1 Building from source tarballs . . . 3

1.4 Configuration . . . 3

1.4.1 Configuration files . . . 3

1.4.2 Configuration syntax . . . 4

1.4.3 Environment variables . . . 4

1.5 Customization . . . 5

1.6 Thanks . . . 5

1.7 Copyright . . . 5

2 Todo List **5**

3 Deprecated List **6**

4 Bug List **6**

5 Data Structure Index **6**

5.1 Data Structures . . . 6

6 File Index **6**

6.1 File List . . . 6

7 Data Structure Documentation **8**

7.1 log4c_appender_type Struct Reference 8

7.1.1 Detailed Description 8

7.2 log4c_buffer_t Struct Reference 8

7.2.1 Detailed Description 9

7.3 log4c_layout_type Struct Reference 9

7.3.1 Detailed Description 9

7.4 log4c_location_info_t Struct Reference 9

7.4.1 Detailed Description 10

7.5 log4c_logging_event_t Struct Reference 10

7.5.1 Detailed Description 10

7.6 log4c_rc_t Struct Reference 11

7.6.1 Detailed Description 11

7.7 log4c_rollingpolicy_type Struct Reference 11

7.7.1 Detailed Description 11

8 File Documentation **12**

8.1 appender.h File Reference 12

8.1.1 Detailed Description 13

8.1.2 Macro Definition Documentation 13

8.1.3 Typedef Documentation 14

8.1.4 Function Documentation 14

8.2 appender_type_mmap.h File Reference 18

8.2.1 Detailed Description 19

8.2.2 Variable Documentation 19

8.3 appender_type_rollingfile.h File Reference 20

8.3.1 Detailed Description 20

8.3.2 Function Documentation 21

8.3.3 Variable Documentation 23

8.4 appender_type_stream.h File Reference 23

8.4.1 Detailed Description 24

8.4.2 Variable Documentation 24
8.5 appender_type_stream2.h File Reference 24
8.5.1 Detailed Description 25
8.5.2 Function Documentation 26
8.5.3 Variable Documentation 27
8.6 appender_type_syslog.h File Reference 27
8.6.1 Detailed Description 28
8.6.2 Variable Documentation 28
8.7 buffer.h File Reference 29
8.7.1 Detailed Description 29
8.8 category.h File Reference 29
8.8.1 Detailed Description 31
8.8.2 Macro Definition Documentation 31
8.8.3 Typedef Documentation 31
8.8.4 Function Documentation 31
8.9 init.h File Reference 42
8.9.1 Detailed Description 42
8.9.2 Function Documentation 43
8.10 layout.h File Reference 43
8.10.1 Detailed Description 44
8.10.2 Macro Definition Documentation 44
8.10.3 Typedef Documentation 45
8.10.4 Function Documentation 45
8.11 layout_type_basic.h File Reference 49
8.11.1 Detailed Description 50
8.12 layout_type_basic_r.h File Reference 50
8.12.1 Detailed Description 51
8.13 layout_type_dated.h File Reference 51
8.13.1 Detailed Description 52
8.14 layout_type_dated_local.h File Reference 52

8.14.1 Detailed Description . . . 53
8.15 layout_type_dated_local_r.h File Reference . . . 53
8.15.1 Detailed Description . . . 54
8.16 layout_type_dated_r.h File Reference . . . 54
8.16.1 Detailed Description . . . 55
8.17 location_info.h File Reference . . . 55
8.17.1 Detailed Description . . . 56
8.17.2 Macro Definition Documentation . . . 56
8.18 logging_event.h File Reference . . . 57
8.18.1 Detailed Description . . . 57
8.18.2 Function Documentation . . . 57
8.19 priority.h File Reference . . . 58
8.19.1 Detailed Description . . . 59
8.19.2 Enumeration Type Documentation . . . 59
8.19.3 Function Documentation . . . 59
8.20 rc.h File Reference . . . 60
8.20.1 Detailed Description . . . 60
8.20.2 Function Documentation . . . 60
8.20.3 Variable Documentation . . . 61
8.21 rollingpolicy.h File Reference . . . 61
8.21.1 Detailed Description . . . 62
8.21.2 Macro Definition Documentation . . . 62
8.21.3 Typedef Documentation . . . 63
8.21.4 Function Documentation . . . 63
8.22 rollingpolicy_type_sizewin.h File Reference . . . 66
8.22.1 Detailed Description . . . 67
8.22.2 Typedef Documentation . . . 67
8.22.3 Function Documentation . . . 67
8.23 version.h File Reference . . . 68
8.23.1 Detailed Description . . . 69
8.23.2 Macro Definition Documentation . . . 69
8.23.3 Function Documentation . . . 69
8.23.4 Variable Documentation . . . 69

Index 71

1 Log4c : Logging for C Library

1.1 Summary

Log4c is a library of C for flexible logging to files, syslog and other destinations. It is modeled after the Log for Java library (`http://jakarta.apache.org/log4j/`), staying as close to their API as is reasonable. Here is a `short introduction` to Log4j which describes the API, and design rationale.

Mark Mendel started a parallel log4c projet with a different philosophy. The design is macro oriented, so much lighter and faster which perfect for kernel development.

Log4c is also available from SourceForge (`http://www.sourceforge.net/projects/log4c/`). This is work in progress.

1.2 Requirements

1.2.1 Platforms

log4c was successfully compiled and run on the following platforms:

- HP-UX release 11.00
- Tru 64 release 4.0F and 5.1
- Red Hat Linux Intel release 7.x, 8, 9
- Red Hat Enterprise Linux 3, 4
- Solaris Intel release 8, 9, 10
- FreeBSD 6.1-RELEASE
- AIX 5.3 (with xlc compiler)
- Mac OS X
- Windows X
- ...and other Linux distributions

log4c should compile and run on the following platforms:

- The BSD family
- MS Windows

1.2.2 Software

The following software is needed to generate log4c:

- GCC 3.0.1+, to generate log4c, but hopefully not to use it.
- doxygen 1.2.13+, a documentation system for C/C++ needed to generate the documentation.
- graphviz, the AT&T Graph Visualization Tools also needed to generate the documentation.

For the moment, log4c uses specific GCC extensions, like **attribute**, so you will need GCC to compile it. This will probably change one day.

1.3 Installation

1.3.1 Building from source tarballs

on SourceForge:

- log4c-1.2.4.tar.gz

The log4c package uses the GNU autotools compilation and installation framework. The following commands should build log4c on the supported platforms:

```
$ gzip -dc log4c-1.2.4.tar.gz | tar tvf -
$ cd log4c-1.2.4/
$ ./configure --prefix=/path/of/installation
$ make
$ make install
```

Checkout the INSTALL file for installation and the generated doxygen documentation for more information.

1.4 Configuration

1.4.1 Configuration files

log4c searches the following files to load its configuration:

- ${LOG4C_RCPATH}/log4crc
- ${HOME}/.log4crc
- ./log4crc

The environment variable LOG4C_RCPATH holds the prefix used for installation.

1.4.2 Configuration syntax

The `log4crc` configuration file uses an XML syntax. The root element is <log4c> and it can be used to control the configuration file version interface with the attribute "`version`". The following 4 elements are supported: <config>, <category>, <appender> and <layout>.

- The <config> element controls the global log4c configuration. It has 3 sub elements. The <nocleanup> flag inhibits the log4c destructors routines. The <bufsize> element sets the buffer size used to format **log4c_logging_event_t** (p. 10) objects. If is set to 0, the allocation is dynamic (the <debug> element is currently unused).

- The <category> element has 3 possible attributes: the category "`name`", the category "`priority`" and the category "`appender`". Future versions will handle multiple appenders per category.

- The <appender> element has 3 possible attributes: the appender "`name`", the appender "`type`", and the appender "`layout`".

- The <layout> element has 2 possible attributes: the layout "`name`" and the layout "`type`".

Here's the `log4crc` configuration file example:

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <!DOCTYPE log4c SYSTEM "">
3
4 <log4c>
5
6         <config>
7                 <bufsize>0</bufsize>
8                 <debug level="0"/>
9                 <nocleanup>0</nocleanup>
10         </config>
11
12         <!-- root category ============================================ -->
13         <category name="root" priority="notice"/>
14
15         <!-- default appenders ====================================== -->
16         <appender name="stdout" type="stream" layout="basic"/>
17         <appender name="stderr" type="stream" layout="dated"/>
18         <appender name="syslog" type="syslog" layout="basic"/>
19
20         <!-- default layouts ======================================== -->
21         <layout name="basic" type="basic"/>
22         <layout name="dated" type="dated"/>
23 </log4c>
```

This initial version of the log4c configuration file syntax is quite different from log4j. XML seemed the best choice to keep the log4j configuration power in a C API.

1.4.3 Environment variables

- `LOG4C_RCPATH` holds the path to the main `log4crc` configuration file
- `LOG4C_PRIORITY` holds the "`root`" category priority
- `LOG4C_APPENDER` holds the "`root`" category appender

1.5 Customization

This section will, one day, briefly describe how to define custom appenders and custom layouts. Be patient or checkout the source.

1.6 Thanks

Mark Mendel for his work on a previous version of log4c.

This project would not have existed without Ceki Gulcu, the creator and maintainer of `Log4j`, nor without Bastiaan Bakker, who initiated me with `Log4Cpp`.

Many thanks to

- Joel Schaubert for many contributions
- Robert Byrne for Windows port and also many contributions
- Olger Warnier for the Mac OS X port
- Jeff Smith for writing a primer on `how to use Log4c effectively`

1.7 Copyright

All software in this package is Copyright (C) 2003-2004 Meiosys `http://www.meiosys.com` and Cedric Le Goater and is distributed under the `LGPL` License. See the `COPYING` file for full legal details.

2 Todo List

File appender.h (p. **12**)

the appender interface needs a better configuration system depending on the layout type. The udata field is a just a trick.

File layout.h (p. **43**)

the layout interface needs a better configuration system depending on the layout type. The udata field is a just a trick.

a pattern layout would be welcomed !!

Global log4c_category_get_chainedpriority (p. **35**) **(const log4c_category_t *a_category)**

the log4c_category_t is designed so that this method executes as quickly as possible. It could even be faster if the set priority was propagated through the children hierarchy of a category.

Global log4c_category_set_appender (p. **41**) **(log4c_category_t *a_category, struct __log4c_appender *a_appender)**

need multiple appenders per category

Class log4c_location_info_t (p. **9**)

this is not used

Global log4c_logging_event_new (p. **58**) **(const char *a_category, int a_priority, const char *a_message)**

need to handle multi-threading (NDC)

3 Deprecated List

Global log4c_appender_type_define (p. 13) (a_type)

This macro, and the static initialialization of appenders in general, is deprecated. Use rather the **log4c_appender_type_set()** (p. 18) function to initialize your appenders before calling **log4c_init()** (p. 43)

Global log4c_layout_type_define (p. 44) (a_type)

This macro, and the static initialialization of layouts in general, is deprecated. Use rather the **log4c_layout_type_set()** (p. 48) function to initialize your appenders before calling **log4c_init()** (p. 43)

4 Bug List

Global log4c_appender_append (p. 14) (log4c_appender_t *a_appender, log4c_logging_event_t (p. 10) *a_event)

is this the right place to open an appender ?

Global log4c_category_get (p. 33) (const char *a_name)

the root category name should be "" not "root". *

5 Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

log4c_appender_type
Log4c appender type class 8

log4c_buffer_t
Buffer object 8

log4c_layout_type
Log4c layout type class 9

log4c_location_info_t
Logging location information 9

log4c_logging_event_t
Logging event object 10

log4c_rc_t
Resource configuration object 11

log4c_rollingpolicy_type
Log4c rollingpolicy type. Defines the interface a specific policy must provide to the rollingfile appender 11

6 File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

appender.h
Implement this interface for your own strategies for printing log statements 12

appender_type_mmap.h
Log4c mmap(2) appender interface 18

appender_type_rollingfile.h
Log4c rolling file appender interface 20

appender_type_stream.h
Log4c stream appender interface 23

appender_type_stream2.h
Log4c stream2 appender interface 24

appender_type_syslog.h
Log4c syslog(3) appender interface 27

buffer.h
Log4c buffer 29

category.h
Central class in the log4c package 29

config-win32.h ??

init.h
Log4c constructors and destructors 42

layout.h
Interface for user specific layout format of log4c_logging_event events 43

layout_type_basic.h
Implement a basic layout 49

layout_type_basic_r.h
Implement a basic_r layout 50

layout_type_dated.h
Implement a dated layout 51

layout_type_dated_local.h
Implement a dated layout with local time 52

layout_type_dated_local_r.h
Implement a dated layout (reentrant) with local time 53

layout_type_dated_r.h
Implement a dated_r layout 54

location_info.h
The internal representation of caller location information 55

logging_event.h
Internal representation of logging events 57

priority.h
The priority class provides importance levels with which one can categorize log messages 58

rc.h
Log4c resource configuration 60

rollingpolicy.h
Log4c rolling policy interface. Defines the interface for managing and providing rolling policies 61

rollingpolicy_type_sizewin.h
Log4c rolling file size-win interface. Log4c ships with (and defaults to) the classic size-window rollover policy: this triggers rollover when files reach a maximum size. The first file in the list is always the current file; when a rollover event occurs files are shifted up by one position in the list–if the number of files in the list has already reached the max then the oldest file is rotated out of the window 66

version.h
Log4c version information 68

7 Data Structure Documentation

7.1 log4c_appender_type Struct Reference

log4c appender type class

`#include <appender.h>`

7.1.1 Detailed Description

log4c appender type class

Attributes description:

- `name` appender type name
- `open`
- `append`
- `close`

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

- **appender.h**

7.2 log4c_buffer_t Struct Reference

buffer object

`#include <buffer.h>`

7.2.1 Detailed Description

buffer object

Attributes description:

- `size` current size of the buffer
- `maxsize` maximum size of the buffer. 0 means no limitation.
- `data` raw data

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

- **buffer.h**

7.3 log4c_layout_type Struct Reference

log4c layout type class

`#include <layout.h>`

7.3.1 Detailed Description

log4c layout type class

Attributes description:

- `name` layout type name
- `format`

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

- **layout.h**

7.4 log4c_location_info_t Struct Reference

logging location information

`#include <location_info.h>`

7.4.1 Detailed Description

logging location information

Attributes description:

- `loc_file` file name
- `loc_line` file line
- `loc_function` function name
- `loc_data` user data

Todo this is not used

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

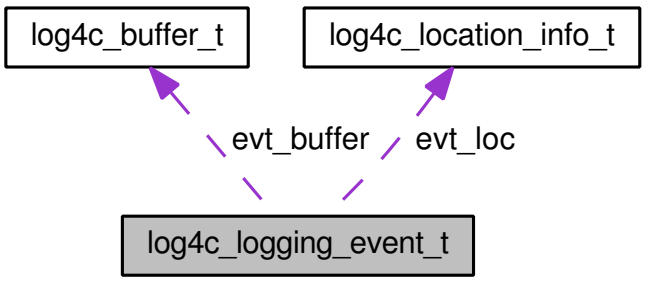
- **location_info.h**

7.5 log4c_logging_event_t Struct Reference

logging event object

`#include <logging_event.h>`

Collaboration diagram for log4c_logging_event_t:



7.5.1 Detailed Description

logging event object

Attributes description:

- `evt_category` category name.
- `evt_priority` priority of logging event.
- `evt_msg` The application supplied message of logging event.
- `evt_buffer` a pre allocated buffer to be used by layouts to format in a multi-thread environment.
- `evt_rendered_msg` The application supplied message after layout format.
- `evt_timestamp` The number of seconds elapsed since the epoch (1/1/1970 00:00:00 UTC) until logging event was created.
- `evt_loc` The event's location information

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

- **logging_event.h**

7.6 log4c_rc_t Struct Reference

resource configuration object

`#include <rc.h>`

7.6.1 Detailed Description

resource configuration object

Attributes description:

- `nocleanup` don't perform memory cleanup in log4c library destructor or in **log4c_fini()** (p. 43)
- `bufsize` maximum logging buffer size. 0 for no limits
- `debug` activate log4c debugging

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

- **rc.h**

7.7 log4c_rollingpolicy_type Struct Reference

log4c rollingpolicy type. Defines the interface a specific policy must provide to the rollingfile appender.

`#include <rollingpolicy.h>`

7.7.1 Detailed Description

log4c rollingpolicy type. Defines the interface a specific policy must provide to the rollingfile appender.

Attributes description:

- `name` rollingpolicy type name
- `init()` init the rollingpolicy
- `is_triggering_event()`
- `rollover()`

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

- **rollingpolicy.h**

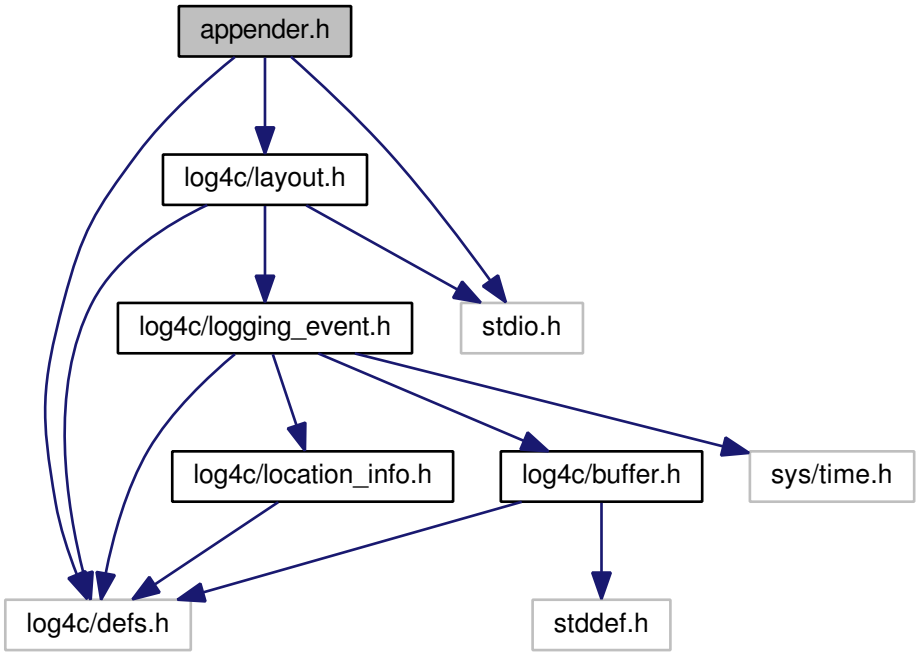
8 File Documentation

8.1 appender.h File Reference

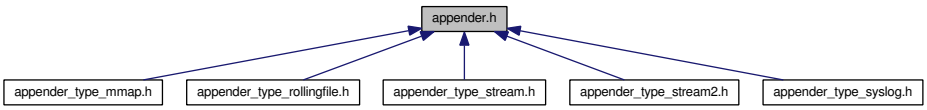
Implement this interface for your own strategies for printing log statements.

```
#include <log4c/defs.h>
#include <log4c/layout.h>
#include <stdio.h>
```

Include dependency graph for appender.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **log4c_appender_type**

  log4c appender type class

Macros

- #define **log4c_appender_type_define**(a_type)

Typedefs

- typedef struct __log4c_appender **log4c_appender_t**
- typedef struct **log4c_appender_type log4c_appender_type_t**

  log4c appender type class

Functions

- const **log4c_appender_type_t** * **log4c_appender_type_get** (const char *a_name)
- const **log4c_appender_type_t** * **log4c_appender_type_set** (const **log4c_appender_type_t** *a_type)
- **log4c_appender_t** * **log4c_appender_get** (const char *a_name)
- **log4c_appender_t** * **log4c_appender_new** (const char *a_name)
- void **log4c_appender_delete** (**log4c_appender_t** *a_appender)
- const char * **log4c_appender_get_name** (const **log4c_appender_t** *a_appender)
- const **log4c_appender_type_t** * **log4c_appender_get_type** (const **log4c_appender_t** *a_appender)
- const **log4c_layout_t** * **log4c_appender_get_layout** (const **log4c_appender_t** *a_appender)
- void * **log4c_appender_get_udata** (const **log4c_appender_t** *a_appender)
- const **log4c_appender_type_t** * **log4c_appender_set_type** (**log4c_appender_t** *a_appender, const **log4c_appender_type_t** *a_type)
- void * **log4c_appender_set_udata** (**log4c_appender_t** *a_appender, void *a_udata)
- const **log4c_layout_t** * **log4c_appender_set_layout** (**log4c_appender_t** *a_appender, const **log4c_↩ layout_t** *a_layout)
- int **log4c_appender_open** (**log4c_appender_t** *a_appender)
- int **log4c_appender_append** (**log4c_appender_t** *a_appender, **log4c_logging_event_t** *a_event)
- int **log4c_appender_close** (**log4c_appender_t** *a_appender)
- void **log4c_appender_print** (const **log4c_appender_t** *a_appender, FILE *a_stream)
- void **log4c_appender_types_free** (void)
- void **log4c_appender_types_print** (FILE *fp)

8.1.1 Detailed Description

Implement this interface for your own strategies for printing log statements.

Todo the appender interface needs a better configuration system depending on the layout type. The udata field is a just a trick.

8.1.2 Macro Definition Documentation

8.1.2.1 #define log4c_appender_type_define(*a_type*)

Helper macro to define static appender types.

Parameters

a_type	the log4c_appender_type_t object to define

Warning

needs GCC support: otherwise this macro does nothing

Deprecated This macro, and the static initialialization of appenders in general, is deprecated. Use rather the **log4c_appender_type_set()** (p. 18) function to initialize your appenders before calling **log4c_init()** (p. 43)

8.1.3 Typedef Documentation

8.1.3.1 typedef struct __log4c_appender log4c_appender_t

log4c appender class

8.1.3.2 typedef struct log4c_appender_type log4c_appender_type_t

log4c appender type class

Attributes description:

- `name` appender type name
- `open`
- `append`
- `close`

8.1.4 Function Documentation

8.1.4.1 int log4c_appender_append (log4c_appender_t * *this,* log4c_logging_event_t * *a_event*)

log in appender specific way.

Parameters

a_appender	the log4c_appender object
a_event	the **log4c_logging_event_t** (p. 10) object to log.

Bug is this the right place to open an appender ?

References log4c_appender_open(), and log4c_layout_format().

8.1.4.2 int log4c_appender_close (log4c_appender_t * *a_appender*)

closes the appender

Parameters

a_appender	the log4c_appender_t object

Returns

zero if successful, -1 otherwise

8.1.4.3 void log4c_appender_delete (log4c_appender_t * a_appender)

Destructor for log4c_appender_t.

References log4c_appender_close().

8.1.4.4 log4c_appender_t* log4c_appender_get (const char * a_name)

Get a pointer to an existing appender.

Parameters

a_name	the name of the appender to return.

Returns

a pointer to an existing appender, or NULL if no appender with the specfied name exists.

References log4c_appender_delete(), log4c_appender_new(), log4c_appender_print(), and log4c_appender_set↩
_udata().

8.1.4.5 const log4c_layout_t* log4c_appender_get_layout (const log4c_appender_t * a_appender)

Parameters

a_appender	the log4c_appender_t object

Returns

the appender layout

8.1.4.6 const char* log4c_appender_get_name (const log4c_appender_t * a_appender)

Parameters

a_appender	the log4c_appender_t object

Returns

the appender name

8.1.4.7 const log4c_appender_type_t* log4c_appender_get_type (const log4c_appender_t * a_appender)

Parameters

a_appender	the log4c_appender_t object

Returns

the appender operations

8.1.4.8 void* log4c_appender_get_udata (const log4c_appender_t * *a_appender*)

Parameters

a_appender	the log4c_appender_t object

Returns

the appender user data

8.1.4.9 log4c_appender_t* log4c_appender_new (const char * *a_name*)

Constructor for log4c_appender_t.

References log4c_appender_type_stream, and log4c_layout_get().

8.1.4.10 int log4c_appender_open (log4c_appender_t * *a_appender*)

opens the appender.

Parameters

a_appender	the log4c_appender_t object

8.1.4.11 void log4c_appender_print (const log4c_appender_t * *a_appender,* FILE * *a_stream*)

prints the appender on a stream

Parameters

a_appender	the log4c_appender_t object
a_stream	the stream

References log4c_layout_get_name().

8.1.4.12 const log4c_layout_t* log4c_appender_set_layout (log4c_appender_t * *a_appender,* const log4c_layout_t * *a_layout*)

sets the appender layout

Parameters

a_appender	the log4c_appender_t object
a_layout	the new appender layout

Returns

the previous appender layout

8.1.4.13 const log4c_appender_type_t* log4c_appender_set_type (log4c_appender_t * *a_appender,* const log4c_appender_type_t * *a_type*)

sets the appender type

Parameters

a_appender	the log4c_appender_t object
a_type	the new appender type

Returns

the previous appender type

8.1.4.14 void* log4c_appender_set_udata (log4c_appender_t * *a_appender,* void * *a_udata*)

sets the appender user data

Parameters

a_appender	the log4c_appender_t object
a_udata	the new appender user data

Returns

the previous appender user data

8.1.4.15 const log4c_appender_type_t* log4c_appender_type_get (const char * *a_name*)

Get a pointer to an existing appender type.

Parameters

a_name	the name of the appender type to return.

Returns

a pointer to an existing appender type, or NULL if no appender type with the specified name exists.

8.1.4.16 const log4c_appender_type_t* log4c_appender_type_set (const log4c_appender_type_t * *a_type*)

Use this function to register an appender type with log4c. Once this is done you may refer to this type by name both programmatically and in the log4c configuration file.

Parameters

a_type	a pointer to the new appender type to set.

Returns

a pointer to the previous appender type of same name.

Example code fragment:

```
1 const log4c_appender_type_t log4c_appender_type_s13_file = {
2   "s13_file",
3   s13_file_open,
4   s13_file_append,
5   s13_file_close,
6 };
7
8  log4c_appender_type_set(&log4c_appender_type_s13_file);
```

8.1.4.17 void log4c_appender_types_free (void)

free all appender types

8.1.4.18 void log4c_appender_types_print (FILE * *fp*)

prints all the current registered appender types on a stream

Parameters

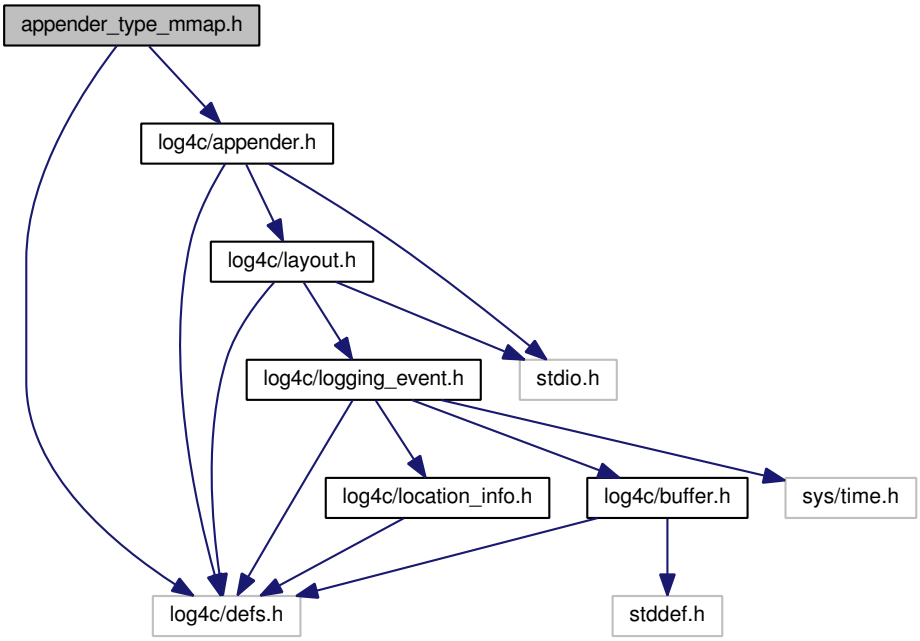
fp	the stream

8.2 appender_type_mmap.h File Reference

Log4c mmap(2) appender interface.

```
#include <log4c/defs.h>
#include <log4c/appender.h>
```

Include dependency graph for appender_type_mmap.h:



Variables

- __LOG4C_BEGIN_DECLS const **log4c_appender_type_t log4c_appender_type_mmap**

8.2.1 Detailed Description

Log4c mmap(2) appender interface.

The mmap appender uses a fixed length memory mapped file for logging. The appender's name is used as the file name which will be opened and mapped to memory at first use. The memory mapped file is then used as a rotating buffer in which logging events are written.

The following examples shows how to define and use mmap appenders.

```
log4c_appender_t* myappender;

myappender = log4c_appender_get("myfile.log");
log4c_appender_set_type(myappender, &log4c_appender_type_mmap);
```

Warning

the file is not created at first use. It should already exist and have a reasonable size, a mutilple of a page size.

8.2.2 Variable Documentation

8.2.2.1 __LOG4C_BEGIN_DECLS const log4c_appender_type_t log4c_appender_type_mmap

Mmap appender type definition.

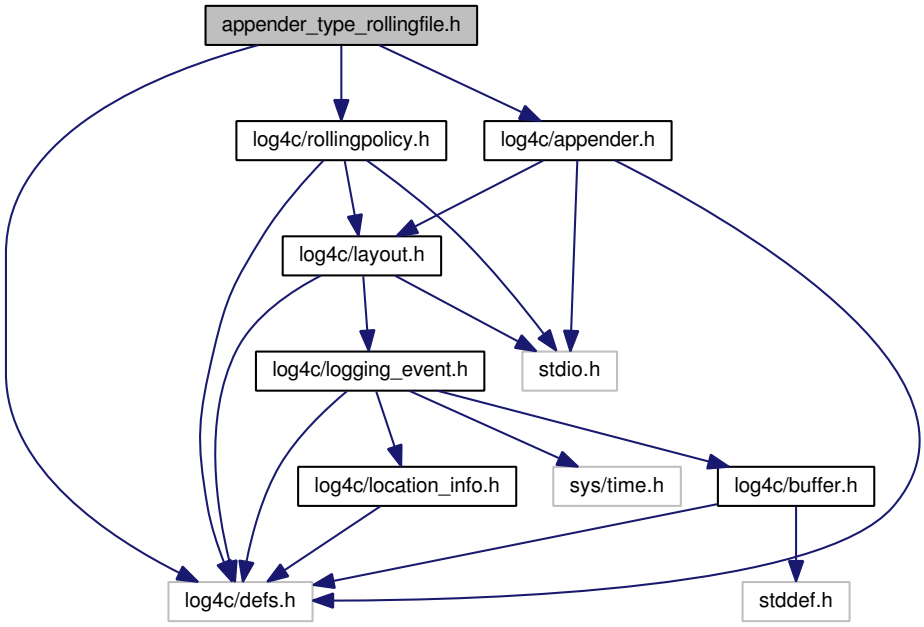
This should be used as a parameter to the **log4c_appender_set_type()** (p. 17) routine to set the type of the appender.

8.3 appender_type_rollingfile.h File Reference

Log4c rolling file appender interface.

```
#include <log4c/defs.h>
#include <log4c/appender.h>
#include <log4c/rollingpolicy.h>
```

Include dependency graph for appender_type_rollingfile.h:



Functions

- rollingfile_udata_t * **rollingfile_make_udata** (void)
- int **rollingfile_udata_set_logdir** (rollingfile_udata_t *rfudatap, const char *logdir)
- int **rollingfile_udata_set_files_prefix** (rollingfile_udata_t *rfudatap, const char *prefix)
- int **rollingfile_udata_set_policy** (rollingfile_udata_t *rfudatap, **log4c_rollingpolicy_t** *policyp)
- const char * **rollingfile_udata_get_logdir** (rollingfile_udata_t *rfudatap)
- const char * **rollingfile_udata_get_files_prefix** (rollingfile_udata_t *rfudatap)
- long **rollingfile_get_current_file_size** (rollingfile_udata_t *rfudatap)

Variables

- __LOG4C_BEGIN_DECLS const **log4c_appender_type_t log4c_appender_type_rollingfile**

8.3.1 Detailed Description

Log4c rolling file appender interface.

The rolling file appender implements a logging mechanism of a list of files up to a maximum number.

The files are written by default to the current directory with logging names folowing the pattern log.1, log.2 etc. These parameters may be changed using the appropriate setter functions.

If the appender fails to open logfiles for writing then the messages are logged to stderr–it will continue to try to open the zero-th file for writing at rollover events so if it succeeds at some point to open that file the messages will start to appear therein and will no longer be sent to stderr.

Switching from logging from one file to the next is referred to as a 'rollover event'.

The policy that determines when a rollover event should happen is called a 'rolling policy'.

A mechanism is provided to allow different rolling policies to be defined.

Log4c ships with (and defaults to) the classic size-window rollover policy: this triggers rollover when files reach a maximum size. The first file in the list is always the current file; when a rollover event occurs files are shifted up by one position in the list–if the number of files in the list has already reached the max then the oldest file is rotated out of the window.

See the documentation in the **rollingpolicy_type_sizewin.h** (p. 66) file for more details on the size-win rollover policy.

8.3.2 Function Documentation

8.3.2.1 long rollingfile_get_current_file_size (rollingfile_udata_t * *rfudatap*)

Get the prefix string in this rolling file appender configuration.

Parameters

rfudatap	the rolling file appender configuration object.

Returns

the current size of the file being logged to.

References log4c_appender_type_rollingfile.

8.3.2.2 rollingfile_udata_t* rollingfile_make_udata (void)

Get a new rolling file appender configuration object.

Returns

a new rolling file appender configuration object, otherwise NULL.

8.3.2.3 const char* rollingfile_udata_get_files_prefix (rollingfile_udata_t * *rfudatap*)

Get the prefix string in this rolling file appender configuration.

Parameters

rfudatap	the rolling file appender configuration object.

Returns

the prefix.

8.3.2.4 const char* rollingfile_udata_get_logdir (rollingfile_udata_t * *rfudatap*)

Get the logging directory in this rolling file appender configuration.

Parameters

rfudatap	the rolling file appender configuration object.

Returns

the logging directory.

8.3.2.5 int rollingfile_udata_set_files_prefix (rollingfile_udata_t * *rfudatap,* const char * *prefix*)

Set the prefix string in this rolling file appender configuration.

Parameters

rfudatap	the rolling file appender configuration object.
prefix	the logging files prfix to use.

Returns

zero if successful, non-zero otherwise.

8.3.2.6 int rollingfile_udata_set_logdir (rollingfile_udata_t * *rfudatap,* const char * *logdir*)

Set the logging directory in this rolling file appender configuration.

Parameters

rfudatap	the rolling file appender configuration object.
logdir	the logging directory to set.

Returns

zero if successful, non-zero otherwise.

8.3.2.7 int rollingfile_udata_set_policy (rollingfile_udata_t * *rfudatap,* log4c_rollingpolicy_t * *policyp*)

Set the rolling policy in this rolling file appender configuration.

Parameters

rfudatap	the rolling file appender configuration object.
policyp	the logging files prfix to use.

Returns

zero if successful, non-zero otherwise.

8.3.3 Variable Documentation

8.3.3.1 __LOG4C_BEGIN_DECLS const log4c_appender_type_t log4c_appender_type_rollingfile

rollingfile appender type definition.

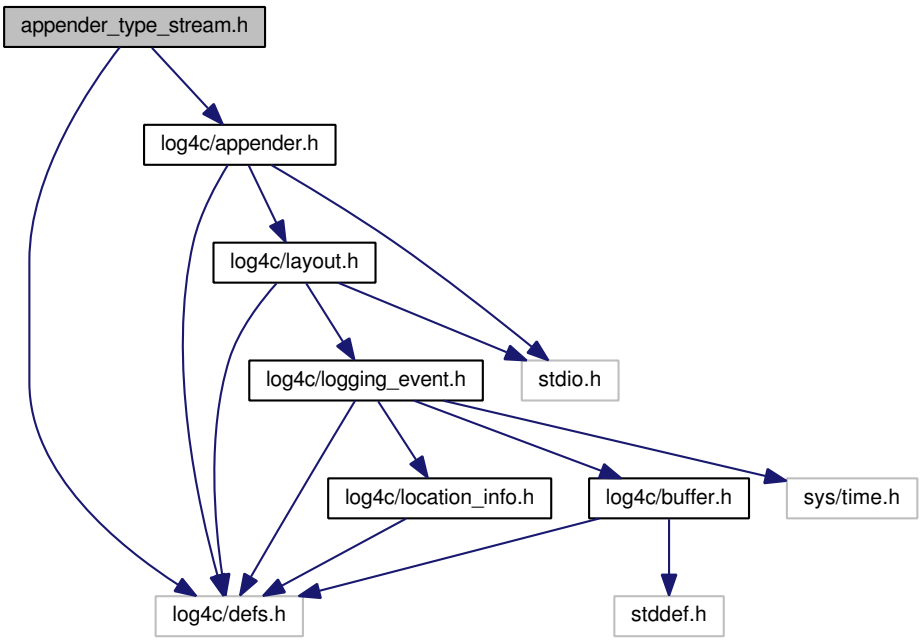
This should be used as a parameter to the **log4c_appender_set_type()** (p. 17) routine to set the type of the appender.

8.4 appender_type_stream.h File Reference

Log4c stream appender interface.

```
#include <log4c/defs.h>
#include <log4c/appender.h>
```

Include dependency graph for appender_type_stream.h:



Variables

- __LOG4C_BEGIN_DECLS const **log4c_appender_type_t log4c_appender_type_stream**

8.4.1 Detailed Description

Log4c stream appender interface.

The stream appender uses a file handle `FILE*` for logging. The appender's name is used as the file name which will be opened at first log. An appender can also be associated to an opened file handle using the **log4c_appender_set_udata()** (p. 17) method to update the appender user data field. In this last case, the appender name has no meaning. 2 default stream appenders are defined: `"stdout"` and `"stderr"`.

The following examples shows how to define and use stream appenders.

- the simple way

```
log4c_appender_t* myappender;

myappender = log4c_appender_get("myfile.log");
log4c_appender_set_type(myappender, &log4c_appender_type_stream);
```

- the sophisticated way

```
log4c_appender_t* myappender;

myappender = log4c_appender_get("myappender");

log4c_appender_set_type(myappender, &log4c_appender_type_stream);
log4c_appender_set_udata(myappender, fopen("myfile.log", "w"));
```

8.4.2 Variable Documentation

8.4.2.1 __LOG4C_BEGIN_DECLS const log4c_appender_type_t log4c_appender_type_stream

Stream appender type definition.

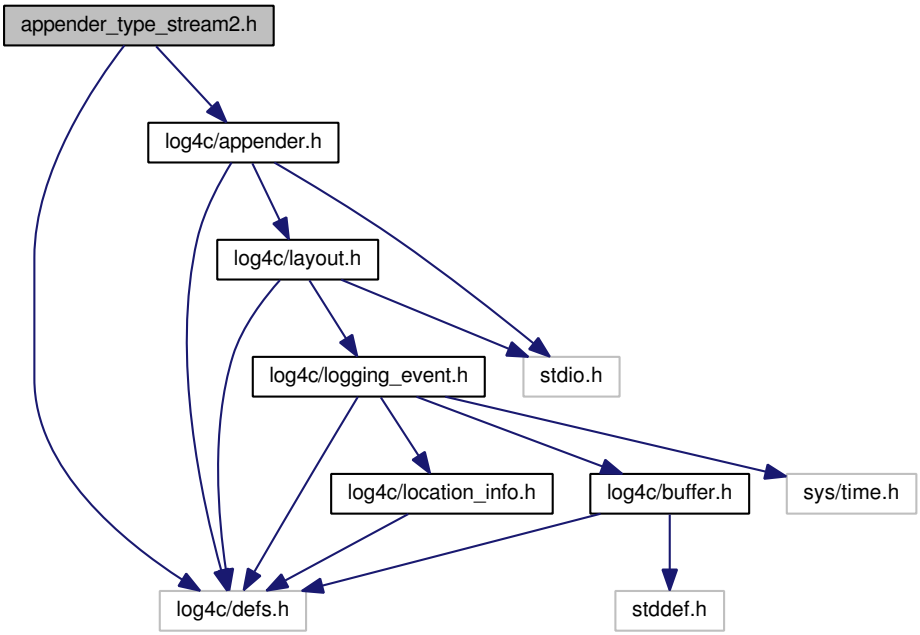
This should be used as a parameter to the **log4c_appender_set_type()** (p. 17) routine to set the type of the appender.

8.5 appender_type_stream2.h File Reference

Log4c stream2 appender interface.

```
#include <log4c/defs.h>
#include <log4c/appender.h>
```

Include dependency graph for appender_type_stream2.h:



Functions

- void **log4c_stream2_set_fp** (**log4c_appender_t** *a_this, FILE *fp)
- FILE * **log4c_stream2_get_fp** (**log4c_appender_t** *a_this)
- void **log4c_stream2_set_flags** (**log4c_appender_t** *a_this, int flags)
- int **log4c_stream2_get_flags** (**log4c_appender_t** *a_this)

Variables

- __LOG4C_BEGIN_DECLS const **log4c_appender_type_t log4c_appender_type_stream2**

8.5.1 Detailed Description

Log4c stream2 appender interface.

The stream2 appender uses a file handle `FILE*` for logging. It can be used with `stdout`, `stderr` or a normal file. It is pretty primitive as it does not do file rotation, or have a maximum configurable file size etc. It improves on the stream appender in a few ways that make it a better starting point for new stream based appenders.

It enhances the stream appender by allowing the default file pointer to be used in buffered or unbuffered mode. Also when you set the file pointer stream2 will not attempt to close it on exit which avoids it fighting with the owner of the file pointer. stream2 is configured via setter functions–the udata is not exposed directly. This means that new options (eg. configure the open mode) could be added to stream2 while maintaining backward compatability.

The appender can be used with default values, for example as follows:

```
log4c_appender_t* myappender;

myappender = log4c_appender_get("/var/logs/mylog.log");
log4c_appender_set_type(myappender,log4c_appender_type_get("stream2"));
```

In this case the appender will be configured automatically with default values:

- the filename is the same as the name of the appender, "`/var/logs/mymlog.log`"
- the file is opened in "w+" mode
- the default system buffer is used (cf; `setbuf()`) in buffered mode

The stream2 appender can be configured by passing it a file pointer to use. In this case you manage the file pointer yourself–open, option setting, closing. If you set the file pointer log4c will not close the file on exiting–you must do this:

```
log4c_appender_t* myappender;
FILE * fp = fopen("myfile.log", "w");

myappender = log4c_appender_get("myappender");
log4c_appender_set_type(myappender, log4c_appender_type_get("stream2"));
log4c_stream2_set_fp(stream2_appender,myfp);
```

The default file pointer can be configured to use unbuffered mode. Buffered mode is typically 25%-50% faster than unbuffered mode but unbuffered mode is useful if your preference is for a more synchronized log file:

```
 log4c_appender_t* myappender;

myappender = log4c_appender_get("/var/logs/mylog.log");
log4c_appender_set_type(myappender,log4c_appender_type_get("stream2"));
log4c_stream2_set_flags(myappender, LOG4C_STREAM2_UNBUFFERED);
```

8.5.2 Function Documentation

8.5.2.1 int log4c_stream2_get_flags (log4c_appender_t * *a_this*)

Get the flags for this appender.

Parameters

this	a pointer to the appender

Returns

the flags for this appender. returns -1 if there was a problem.

8.5.2.2 FILE* log4c_stream2_get_fp (log4c_appender_t * *a_this*)

Get the file pointer for this appender.

Parameters

this	a pointer to the appender

Returns

the file pointer for this appender. If there's a problem returns NULL.

8.5.2.3 void log4c_stream2_set_flags (log4c_appender_t * *a_this*, int *flags*)

Set the flags for this appender.

Parameters

this	a pointer to the appender
flags	ar teh flags to set. These will overwrite the existing flags. Currently supported flags: LOG4C_STREAM2_UNBUFFERED

References log4c_appender_type_stream2.

8.5.2.4 void log4c_stream2_set_fp (log4c_appender_t * *a_this*, FILE * *fp*)

Set the file pointer for this appender.

Parameters

this	a pointer to the appender
fp	the file pointer this appender will use. The caller is responsible for managing the file pointer (open, option setting, closing).

8.5.3 Variable Documentation

8.5.3.1 __LOG4C_BEGIN_DECLS const log4c_appender_type_t log4c_appender_type_stream2

Stream2 appender type definition.

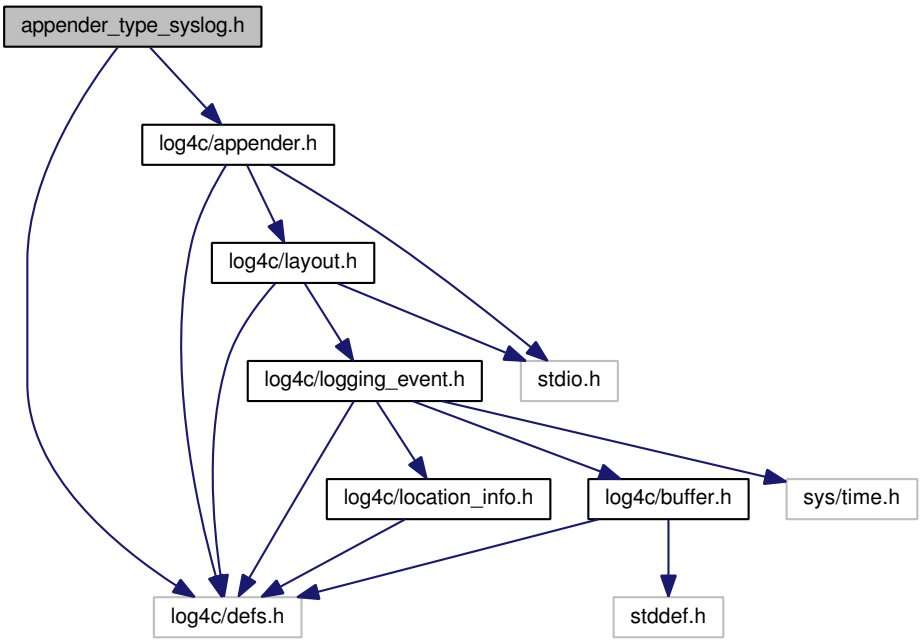
This should be used as a parameter to the **log4c_appender_set_type()** (p. 17) routine to set the type of the appender.

8.6 appender_type_syslog.h File Reference

Log4c syslog(3) appender interface.

```
#include <log4c/defs.h>
#include <log4c/appender.h>
```

Include dependency graph for appender_type_syslog.h:



Variables

- __LOG4C_BEGIN_DECLS const **log4c_appender_type_t log4c_appender_type_syslog**

8.6.1 Detailed Description

Log4c syslog(3) appender interface.

The syslog appender uses the syslog(3) interface for logging. The log4c priorities are mapped to the syslog priorities and the appender name is used as a syslog identifier. 1 default syslog appender is defined: "`syslog`".

The following examples shows how to define and use syslog appenders.

```
log4c_appender_t* myappender;

myappender = log4c_appender_get("myappender");
log4c_appender_set_type(myappender, &log4c_appender_type_syslog);
```

8.6.2 Variable Documentation

8.6.2.1 __LOG4C_BEGIN_DECLS const log4c_appender_type_t log4c_appender_type_syslog

Syslog appender type definition.

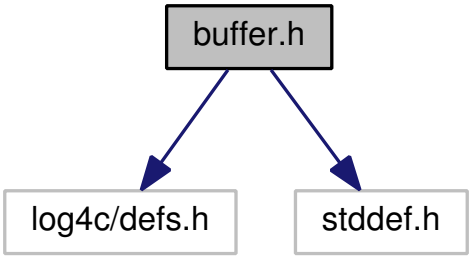
This should be used as a parameter to the **log4c_appender_set_type()** (p. 17) routine to set the type of the appender.

8.7 buffer.h File Reference

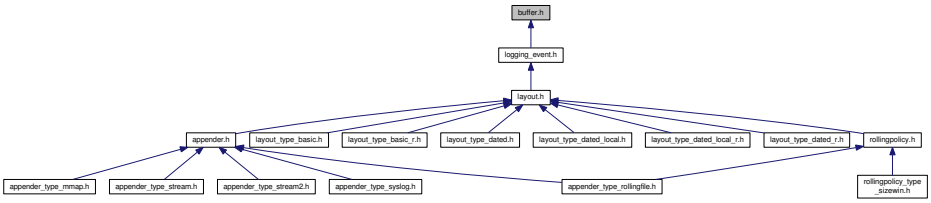
log4c buffer

```
#include <log4c/defs.h>
#include <stddef.h>
```

Include dependency graph for buffer.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **log4c_buffer_t**

  buffer object

8.7.1 Detailed Description

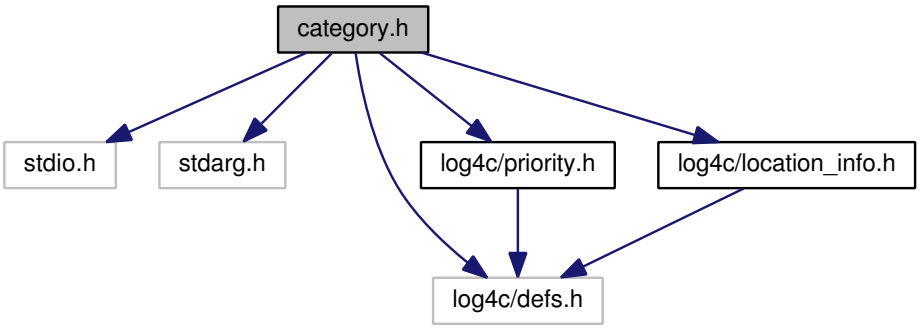
log4c buffer

8.8 category.h File Reference

central class in the log4c package.

```
#include <stdio.h>
#include <stdarg.h>
#include <log4c/defs.h>
#include <log4c/priority.h>
#include <log4c/location_info.h>
```

Include dependency graph for category.h:



Macros

- #define **log4c_category_define**(a_category, a_name)

Typedefs

- typedef struct __log4c_category **log4c_category_t**

Functions

- **log4c_category_t** * **log4c_category_get** (const char *a_name)
- int **log4c_category_list** (**log4c_category_t** **a_cats, int a_ncats)
- **log4c_category_t** * **log4c_category_new** (const char *a_name)
- void **log4c_category_delete** (**log4c_category_t** *a_category)
- const char * **log4c_category_get_name** (const **log4c_category_t** *a_category)
- const struct __log4c_appender * **log4c_category_get_appender** (const **log4c_category_t** *a_category)
- int **log4c_category_get_additivity** (const **log4c_category_t** *a_category)
- int **log4c_category_get_priority** (const **log4c_category_t** *a_category)
- int **log4c_category_get_chainedpriority** (const **log4c_category_t** *a_category)
- const struct __log4c_appender * **log4c_category_set_appender** (**log4c_category_t** *a_category, struct __log4c_appender *a_appender)
- int **log4c_category_set_priority** (**log4c_category_t** *a_category, int a_priority)
- int **log4c_category_set_additivity** (**log4c_category_t** *a_category, int a_additivity)
- void **log4c_category_print** (const **log4c_category_t** *a_category, FILE *a_stream)
- static int **log4c_category_is_priority_enabled** (const **log4c_category_t** *a_category, int a_priority)
- static int **log4c_category_is_fatal_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_alert_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_crit_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_error_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_warn_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_notice_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_info_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_debug_enabled** (const **log4c_category_t** *a_category)
- static int **log4c_category_is_trace_enabled** (const **log4c_category_t** *a_category)
- static LOG4C_INLINE void **log4c_category_log** (const **log4c_category_t** *a_category, int a_priority, const char *a_format,...)
- static LOG4C_INLINE void **log4c_category_log_locinfo** (const **log4c_category_t** *a_category, const **log4c_location_info_t** *a_locinfo, int a_priority, const char *a_format,...)

- static LOG4C_INLINE void **log4c_category_fatal** (const **log4c_category_t** *a_category, const char *a_↩ format,...)
- static LOG4C_INLINE void **log4c_category_alert** (const **log4c_category_t** *a_category, const char *a_↩ format,...)
- static LOG4C_INLINE void **log4c_category_crit** (const **log4c_category_t** *a_category, const char *a_↩ format,...)
- static LOG4C_INLINE void **log4c_category_error** (const **log4c_category_t** *a_category, const char *a_↩ format,...)
- static LOG4C_INLINE void **log4c_category_warn** (const **log4c_category_t** *a_category, const char *a_↩ format,...)
- static LOG4C_INLINE void **log4c_category_notice** (const **log4c_category_t** *a_category, const char *a↩ _format,...)
- static LOG4C_INLINE void **log4c_category_info** (const **log4c_category_t** *a_category, const char *a_↩ format,...)
- static LOG4C_INLINE void **log4c_category_debug** (const **log4c_category_t** *a_category, const char *a↩ _format,...)
- static LOG4C_INLINE void **__log4c_category_trace** (const **log4c_category_t** *a_category, const char *a↩ _format,...)

8.8.1 Detailed Description

central class in the log4c package.

One of the distintive features of log4j (and hence log4c) are hierarchical categories and their evaluation.

8.8.2 Macro Definition Documentation

8.8.2.1 #define log4c_category_define(*a_category, a_name*)

Helper macro to define static categories.

Parameters

a_category	the log4c_category_t pointer name
a_name	the category name

8.8.3 Typedef Documentation

8.8.3.1 typedef struct __log4c_category log4c_category_t

log4c category class

8.8.4 Function Documentation

8.8.4.1 static LOG4C_INLINE void __log4c_category_trace (const log4c_category_t * *a_category,* const char * *a_format,* ...) [static]

Log a message with trace priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_TRACE.

8.8.4.2 static LOG4C_INLINE void log4c_category_alert (const log4c_category_t * *a_category*, const char * *a_format*, ...) [static]

Log a message with alert priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_ALERT.

8.8.4.3 static LOG4C_INLINE void log4c_category_crit (const log4c_category_t * *a_category*, const char * *a_format*, ...) [static]

Log a message with crit priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_CRIT.

8.8.4.4 static LOG4C_INLINE void log4c_category_debug (const log4c_category_t * *a_category*, const char * *a_format*, ...) [static]

Log a message with debug priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_DEBUG.

8.8.4.5 void log4c_category_delete (log4c_category_t * a_category)

Destructor for a log4c_category_t.

Parameters

a_category	the log4c_category_t object

8.8.4.6 static LOG4C_INLINE void log4c_category_error (const log4c_category_t * a_category, const char * a_format, ...) [static]

Log a message with error priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_ERROR.

8.8.4.7 static LOG4C_INLINE void log4c_category_fatal (const log4c_category_t * a_category, const char * a_format, ...) [static]

Log a message with fatal priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_FATAL.

8.8.4.8 log4c_category_t* log4c_category_get (const char * a_name)

Instantiate a log4c_category_t with name *name*. This method does not set priority of the category which is by default `LOG4C_PRIORITY_NOTSET`.

Parameters

a_name	The name of the category to retrieve.

Bug the root category name should be "" not "root". *

References log4c_category_delete(), log4c_category_new(), and log4c_category_print().

8.8.4.9 int log4c_category_get_additivity (const log4c_category_t * *a_category*)

Get the additivity flag for this log4c_category_t..

Parameters

a_category	the log4c_category_t object

Returns

the category additivity

8.8.4.10 const struct __log4c_appender* log4c_category_get_appender (const log4c_category_t * a_category)

Returns the Appender for this log4c_category_t, or NULL if no Appender has been set.

Parameters

a_category	the log4c_category_t object

Returns

The Appender.

8.8.4.11 int log4c_category_get_chainedpriority (const log4c_category_t * a_category)

Starting from this category, search the category hierarchy for a set priority and return it. Otherwise, return the priority of the root category.

Parameters

a_category	the log4c_category_t object

Todo the log4c_category_t is designed so that this method executes as quickly as possible. It could even be faster if the set priority was propagated through the children hierarchy of a category.

References LOG4C_PRIORITY_NOTSET, and LOG4C_PRIORITY_UNKNOWN.

8.8.4.12 const char* log4c_category_get_name (const log4c_category_t * a_category)

Return the category name.

Parameters

a_category	the log4c_category_t object

Returns

the category name.

8.8.4.13 int log4c_category_get_priority (const log4c_category_t * a_category)

Returns the assigned Priority, if any, for this log4c_category_t.

Parameters

a_category	the log4c_category_t object

Returns

Priority - the assigned Priority, can be LOG4C_PRIORITY_NOTSET

References LOG4C_PRIORITY_UNKNOWN.

8.8.4.14 static LOG4C_INLINE void log4c_category_info (const log4c_category_t * a_category, const char * a_format, ...) [static]

Log a message with info priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_INFO.

8.8.4.15 static int log4c_category_is_alert_enabled (const log4c_category_t * a_category) [inline],[static]

Return true if the category will log messages with priority LOG4C_PRIORITY_ALERT.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_ALERT.

8.8.4.16 static int log4c_category_is_crit_enabled (const log4c_category_t * a_category) [inline],[static]

Return true if the category will log messages with priority LOG4C_PRIORITY_CRIT.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_CRIT.

8.8.4.17 static int log4c_category_is_debug_enabled (const log4c_category_t * *a_category*) [inline], [static]

Return true if the category will log messages with priority LOG4C_PRIORITY_DEBUG.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_DEBUG.

8.8.4.18 static int log4c_category_is_error_enabled (const log4c_category_t * *a_category*) [inline], [static]

Return true if the category will log messages with priority LOG4C_PRIORITY_ERROR.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_ERROR.

8.8.4.19 static int log4c_category_is_fatal_enabled (const log4c_category_t * *a_category*) [inline], [static]

Return true if the category will log messages with priority LOG4C_PRIORITY_FATAL.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_FATAL.

8.8.4.20 static int log4c_category_is_info_enabled (const log4c_category_t * *a_category*) [inline],[static]

Return true if the category will log messages with priority LOG4C_PRIORITY_INFO.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_INFO.

8.8.4.21 static int log4c_category_is_notice_enabled (const log4c_category_t * *a_category*) [inline], [static]

Return true if the category will log messages with priority LOG4C_PRIORITY_NOTICE.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_NOTICE.

8.8.4.22 static int log4c_category_is_priority_enabled (const log4c_category_t * *a_category,* int *a_priority*) [inline],[static]

Returns true if the chained priority of the log4c_category_t is equal to or higher than given priority.

Parameters

a_category	the log4c_category_t object
a_priority	The priority to compare with.

Returns

whether logging is enable for this priority.

References log4c_category_get_chainedpriority().

8.8.4.23 static int log4c_category_is_trace_enabled (const log4c_category_t * *a_category*) [inline],[static]

Return true if the category will log messages with priority LOG4C_PRIORITY_TRACE.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), LOG4C_LOCATION_INFO_INITIALIZER, and LOG4C_PRIO↩ RITY_TRACE.

8.8.4.24 static int log4c_category_is_warn_enabled (const log4c_category_t * *a_category*) [inline],[static]

Return true if the category will log messages with priority `LOG4C_PRIORITY_WARN`.

Parameters

a_category	the log4c_category_t object

Returns

Whether the category will log.

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_WARN.

8.8.4.25 int log4c_category_list (log4c_category_t ** *a_cats*, int *a_ncats*)

Fill in an array with the log4c categories.

Parameters

a_cats	array of categories that will be filled
a_ncats	number of categories in the array

Returns

-1 if it fails or the number of available categories in log4c.

8.8.4.26 static LOG4C_INLINE void log4c_category_log (const log4c_category_t * *a_category*, int *a_priority*, const char * *a_format*, ...) [static]

Log a message with the specified priority.

Parameters

a_category	the log4c_category_t object
a_priority	The priority of this log message.
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled().

8.8.4.27 static LOG4C_INLINE void log4c_category_log_locinfo (const log4c_category_t * *a_category*, const log4c_location_info_t * *a_locinfo*, int *a_priority*, const char * *a_format*, ...) [static]

Log a message with the specified priority and a user location info.

Parameters

a_category	the log4c_category_t object
a_locinfo	a user location info
a_priority	The priority of this log message.
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled().

8.8.4.28 log4c_category_t* log4c_category_new (const char * *a_name*)

Constructor for a log4c_category_t.

Parameters

a_name	the category name

Returns

a log4c_category object

Warning

this method should not be called directly. You should use the **log4c_category_get()** (p. 33) method in order to preserve the categories hierarchy.

References log4c_category_get(), and LOG4C_PRIORITY_NOTSET.

8.8.4.29 static LOG4C_INLINE void log4c_category_notice (const log4c_category_t * *a_category*, const char * *a_format*, ...) [static]

Log a message with notice priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_NOTICE.

8.8.4.30 void log4c_category_print (const log4c_category_t * *a_category,* FILE * *a_stream*)

prints the log4c_category_t object on a stream

Parameters

a_category	the log4c_category_t object
a_stream	The stream

References log4c_appender_append(), log4c_appender_get_name(), log4c_category_get_name(), log4c_priority_to_string(), and log4c_rc.

8.8.4.31 int log4c_category_set_additivity (log4c_category_t * *a_category,* int *a_additivity*)

Sets a new additivity flag for this category.

Parameters

a_category	the log4c_category_t object
a_additivity	the new category additivity

Returns

the previous category additivity

8.8.4.32 const struct __log4c_appender* log4c_category_set_appender (log4c_category_t * *this,* log4c_appender_t * *a_appender*)

Sets a new appender for this category.

Parameters

a_category	the log4c_category_t object
a_appender	the new category appender

Returns

the previous category appender

Todo need multiple appenders per category

8.8.4.33 int log4c_category_set_priority (log4c_category_t * *a_category,* int *a_priority*)

Sets a new priority of this category.

Parameters

a_category	the log4c_category_t object
a_priority	the new priority to set. Use LOG4C_PRIORITY_NOTSET to let the category use its parents priority as effective priority.

Returns

the previous category priority

References LOG4C_PRIORITY_UNKNOWN.

8.8.4.34 static LOG4C_INLINE void log4c_category_warn (const log4c_category_t * *a_category,* const char * *a_format,* ...) [static]

Log a message with warn priority.

Parameters

a_category	the log4c_category_t object
a_format	Format specifier for the string to write in the log file.
...	The arguments for a_format

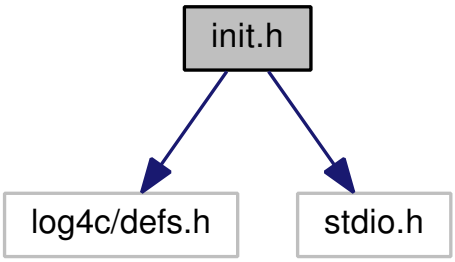
References log4c_category_is_priority_enabled(), and LOG4C_PRIORITY_WARN.

8.9 init.h File Reference

log4c constructors and destructors

```
#include <log4c/defs.h>
#include <stdio.h>
```

Include dependency graph for init.h:



Functions

- int **log4c_init** (void)
- int **log4c_fini** (void)

8.9.1 Detailed Description

log4c constructors and destructors

8.9.2 Function Documentation

8.9.2.1 int log4c_fini (void)

destructor

Returns

0 for success

References log4c_appender_types_free(), log4c_appender_types_print(), log4c_init(), log4c_layout_types_free(), log4c_layout_types_print(), and log4c_rc.

8.9.2.2 int log4c_init (void)

constructor

Returns

0 for success

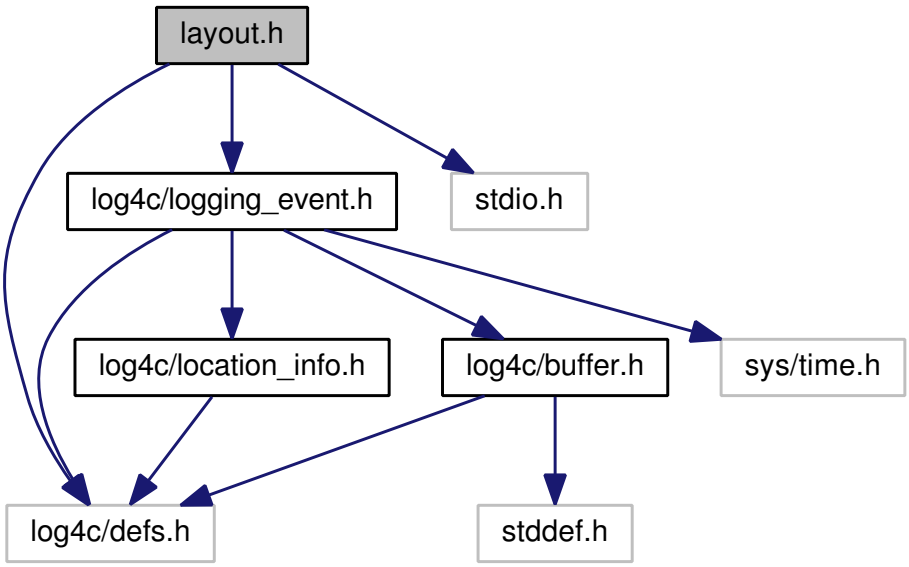
References log4c_appender_get(), log4c_appender_type_set(), log4c_category_get(), log4c_category_set_↩ appender(), log4c_category_set_priority(), log4c_layout_type_set(), log4c_load(), log4c_priority_to_int(), log4c_rc, and log4c_rollingpolicy_type_set().

8.10 layout.h File Reference

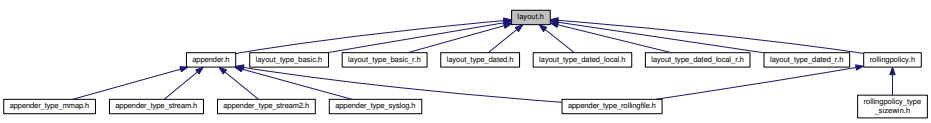
Interface for user specific layout format of log4c_logging_event events.

```
#include <log4c/defs.h>
#include <log4c/logging_event.h>
#include <stdio.h>
```

Include dependency graph for layout.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **log4c_layout_type**

  log4c layout type class

Macros

- #define **log4c_layout_type_define**(a_type)

Typedefs

- typedef struct __log4c_layout **log4c_layout_t**
- typedef struct **log4c_layout_type** **log4c_layout_type_t**

  log4c layout type class

Functions

- const **log4c_layout_type_t** ∗ **log4c_layout_type_get** (const char ∗a_name)
- const **log4c_layout_type_t** ∗ **log4c_layout_type_set** (const **log4c_layout_type_t** ∗a_type)
- **log4c_layout_t** ∗ **log4c_layout_get** (const char ∗a_name)
- **log4c_layout_t** ∗ **log4c_layout_new** (const char ∗a_name)
- void **log4c_layout_delete** (**log4c_layout_t** ∗a_layout)
- const char ∗ **log4c_layout_get_name** (const **log4c_layout_t** ∗a_layout)
- const **log4c_layout_type_t** ∗ **log4c_layout_get_type** (const **log4c_layout_t** ∗a_layout)
- const **log4c_layout_type_t** ∗ **log4c_layout_set_type** (**log4c_layout_t** ∗a_layout, const **log4c_layout_↩ type_t** ∗a_type)
- void ∗ **log4c_layout_get_udata** (const **log4c_layout_t** ∗a_layout)
- void ∗ **log4c_layout_set_udata** (**log4c_layout_t** ∗a_layout, void ∗a_udata)
- const char ∗ **log4c_layout_format** (const **log4c_layout_t** ∗a_layout, const **log4c_logging_event_t** ∗a_↩ event)
- void **log4c_layout_print** (const **log4c_layout_t** ∗a_layout, FILE ∗a_stream)
- void **log4c_layout_types_free** (void)
- void **log4c_layout_types_print** (FILE ∗fp)

8.10.1 Detailed Description

Interface for user specific layout format of log4c_logging_event events.

Todo the layout interface needs a better configuration system depending on the layout type. The udata field is a just a trick.

Todo a pattern layout would be welcomed !!

8.10.2 Macro Definition Documentation

8.10.2.1 #define log4c_layout_type_define(*a_type*)

Helper macro to define static layout types.

Parameters

a_type	the log4c_layout_type_t object to define

Warning

needs GCC support: otherwise this macro does nothing

Deprecated This macro, and the static initialialization of layouts in general, is deprecated. Use rather the **log4c_layout_type_set()** (p. 48) function to initialize your appenders before calling **log4c_init()** (p. 43)

8.10.3 Typedef Documentation

8.10.3.1 typedef struct __log4c_layout log4c_layout_t

log4c layout class

8.10.3.2 typedef struct log4c_layout_type log4c_layout_type_t

log4c layout type class

Attributes description:

- `name` layout type name
- `format`

8.10.4 Function Documentation

8.10.4.1 void log4c_layout_delete (log4c_layout_t * *a_layout*)

Destructor for layout.

8.10.4.2 const char* log4c_layout_format (const log4c_layout_t * *a_layout*, const log4c_logging_event_t * *a_event*)

format a log4c_logging_event events to a string.

Parameters

a_layout	the log4c_layout_t object
a_event	a logging_event_t object

Returns

an appendable string.

8.10.4.3 log4c_layout_t* log4c_layout_get (const char * *a_name*)

Get a pointer to an existing layout.

Parameters

a_name	the name of the layout to return.

Returns

a pointer to an existing layout, or NULL if no layout with the specfied name exists.

References log4c_layout_delete(), log4c_layout_new(), log4c_layout_print(), and log4c_layout_set_type().

8.10.4.4 const char* log4c_layout_get_name (const log4c_layout_t * *a_layout*)

Parameters

a_layout	the log4c_layout_t object

Returns

the layout name

8.10.4.5 const log4c_layout_type_t* log4c_layout_get_type (const log4c_layout_t * *a_layout*)

Parameters

a_layout	the log4c_layout_t object

Returns

a log4c_layout_type_t object

8.10.4.6 void* log4c_layout_get_udata (const log4c_layout_t * *a_layout*)

Parameters

a_layout	the log4c_layout_t object

Returns

the layout user data

8.10.4.7 log4c_layout_t* log4c_layout_new (const char * *a_name*)

Constructor for layout.

8.10.4.8 void log4c_layout_print (const log4c_layout_t * *a_layout*, FILE * *a_stream*)

prints the layout on a stream

Parameters

a_layout	the log4c_layout_t object
a_stream	the stream

8.10.4.9 const log4c_layout_type_t* log4c_layout_set_type (log4c_layout_t * *a_layout*, const log4c_layout_type_t * *a_type*)

sets the layout type

Parameters

a_layout	the log4c_layout_t object
a_type	the new layout type

Returns

the previous layout type

8.10.4.10 void* log4c_layout_set_udata (log4c_layout_t * *a_layout*, void * *a_udata*)

sets the layout user data

Parameters

a_layout	the log4c_layout_t object
a_udata	the new layout user data

Returns

the previous layout user data

8.10.4.11 const log4c_layout_type_t* log4c_layout_type_get (const char * *a_name*)

Get a pointer to an existing layout type.

Parameters

a_name	the name of the layout type to return.

Returns

a pointer to an existing layout type, or NULL if no layout type with the specified name exists.

8.10.4.12 const log4c_layout_type_t* log4c_layout_type_set (const log4c_layout_type_t * *a_type*)

Use this function to register a layout type with log4c. Once this is done you may refer to this type by name both programatically and in the log4c configuration file.

Parameters

a_type	a pointer to the new layout type to set.

Returns

a pointer to the previous layout type of same name.

Example code fragment:

```
1 const log4c_layout_type_t log4c_layout_type_xml = {
2    "s13_xml",
3    xml_format,
4 };
5
6 log4c_layout_type_set(&log4c_layout_type_xml);
```

8.10.4.13 void log4c_layout_types_free (void)

free all layout types

8.10.4.14 void log4c_layout_types_print (FILE * *fp*)

prints all the current registered layout types on a stream

Parameters

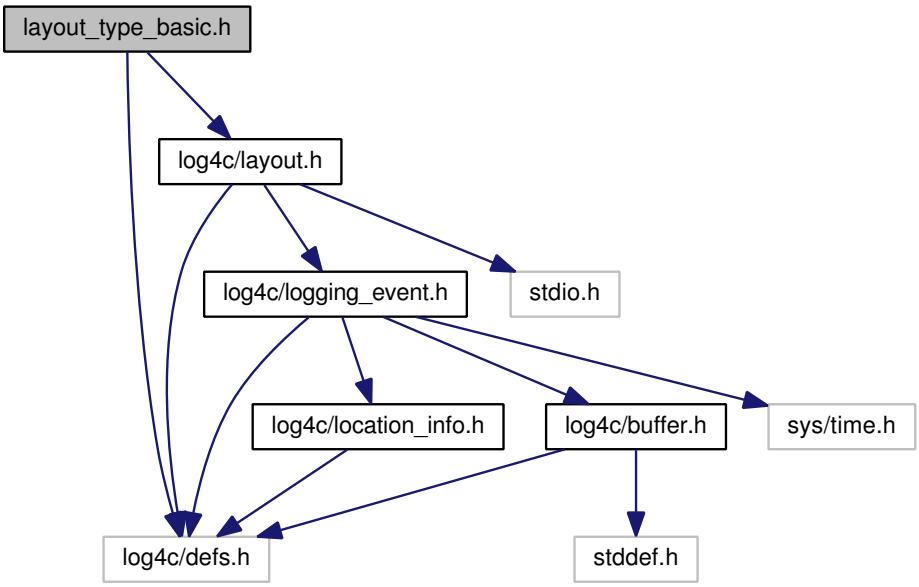
fp	the stream

8.11 layout_type_basic.h File Reference

Implement a basic layout.

```
#include <log4c/defs.h>
#include <log4c/layout.h>
```

Include dependency graph for layout_type_basic.h:



8.11.1 Detailed Description

Implement a basic layout.

In `log4j.PatternLayout` conventions, the basic layout has the following conversion pattern: "`%P %c - %m\n`".

Where

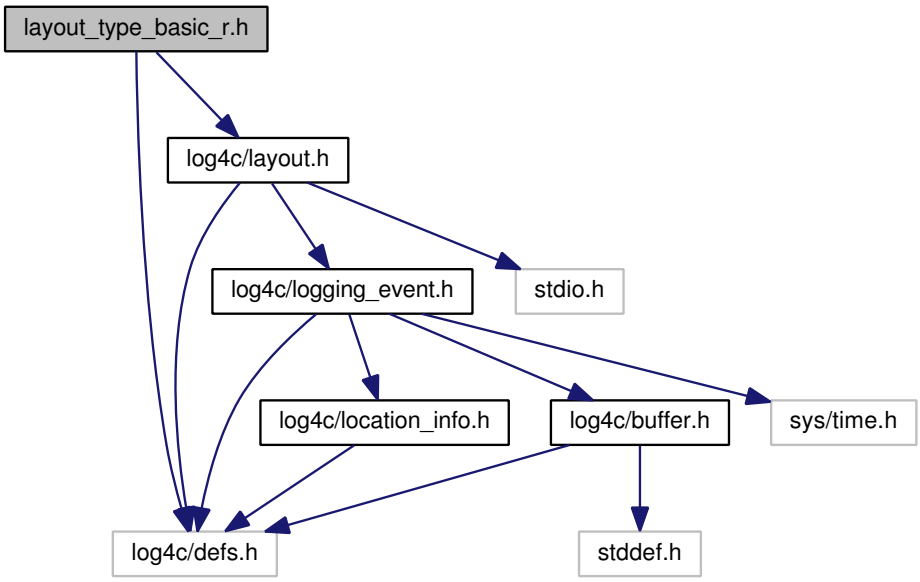
- "`%P`" is the priority of the logging event
- "`%c`" is the category of the logging event
- "`%m`" is the application supplied message associated with the logging event

8.12 layout_type_basic_r.h File Reference

Implement a basic_r layout.

```
#include <log4c/defs.h>
#include <log4c/layout.h>
```

Include dependency graph for layout_type_basic_r.h:



8.12.1 Detailed Description

Implement a basic_r layout.

In `log4j.PatternLayout` conventions, the basic_r layout has the following conversion pattern: "`%P %c - %m\n`".

Where

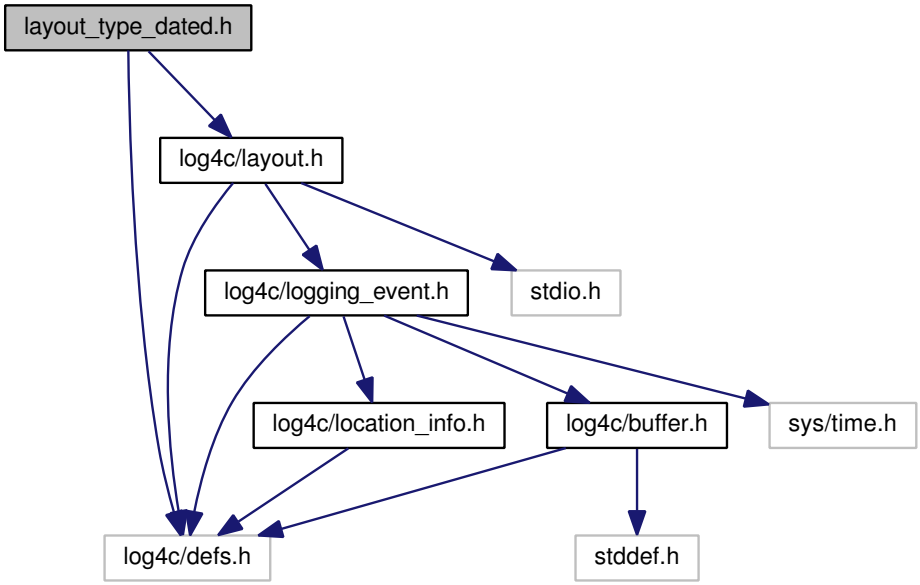
- "`%P`" is the priority of the logging event
- "`%c`" is the category of the logging event
- "`%m`" is the application supplied message associated with the logging event

8.13 layout_type_dated.h File Reference

Implement a dated layout.

```
#include <log4c/defs.h>
#include <log4c/layout.h>
```

Include dependency graph for layout_type_dated.h:



8.13.1 Detailed Description

Implement a dated layout.

In `log4j.PatternLayout` conventions, the dated layout has the following conversion pattern: `"%d %P %c - %m\n"`.

Where

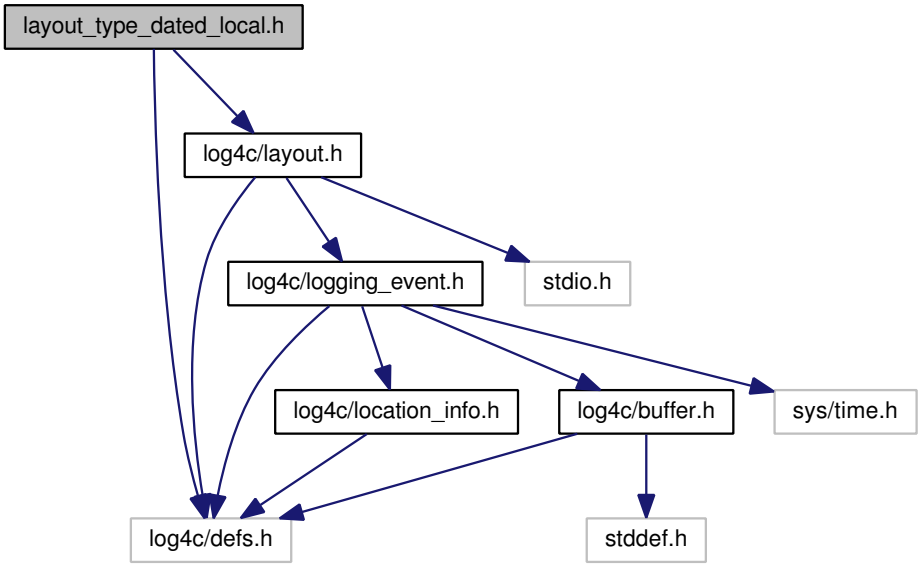
- `"%d"` is the date of the logging event
- `"%P"` is the priority of the logging event
- `"%c"` is the category of the logging event
- `"%m"` is the application supplied message associated with the logging event

8.14 layout_type_dated_local.h File Reference

Implement a dated layout with local time.

```
#include <log4c/defs.h>
#include <log4c/layout.h>
```

Include dependency graph for layout_type_dated_local.h:



8.14.1 Detailed Description

Implement a dated layout with local time.

In `log4j.PatternLayout` conventions, the dated layout has the following conversion pattern: "`%d %P %c - %m\n`".

Where

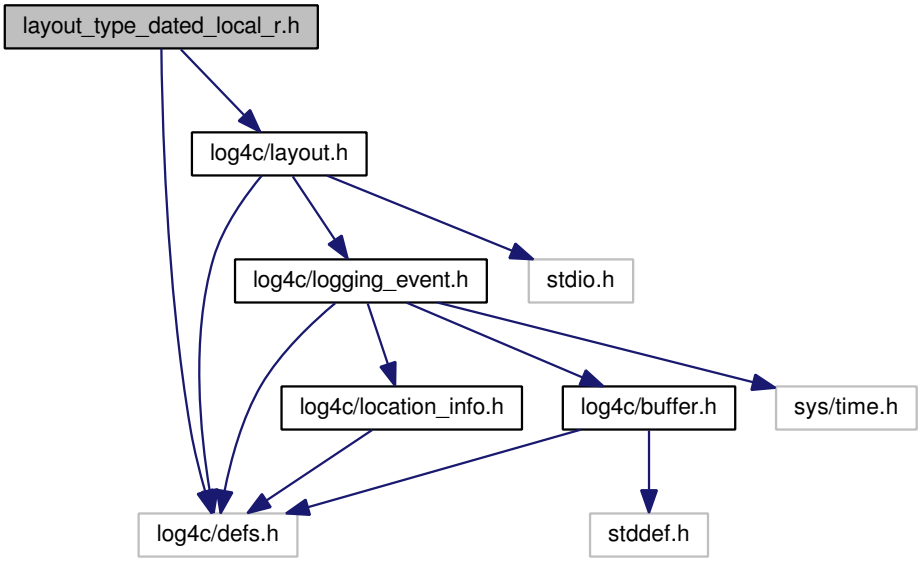
- "`%d`" is the date of the logging event
- "`%P`" is the priority of the logging event
- "`%c`" is the category of the logging event
- "`%m`" is the application supplied message associated with the logging event

8.15 layout_type_dated_local_r.h File Reference

Implement a dated layout (reentrant) with local time.

```
#include <log4c/defs.h>
#include <log4c/layout.h>
```

Include dependency graph for layout_type_dated_local_r.h:



8.15.1 Detailed Description

Implement a dated layout (reentrant) with local time.

In `log4j.PatternLayout` conventions, the dated_local_r layout has the following conversion pattern: "`%d %P %c - %m\n`".

Where

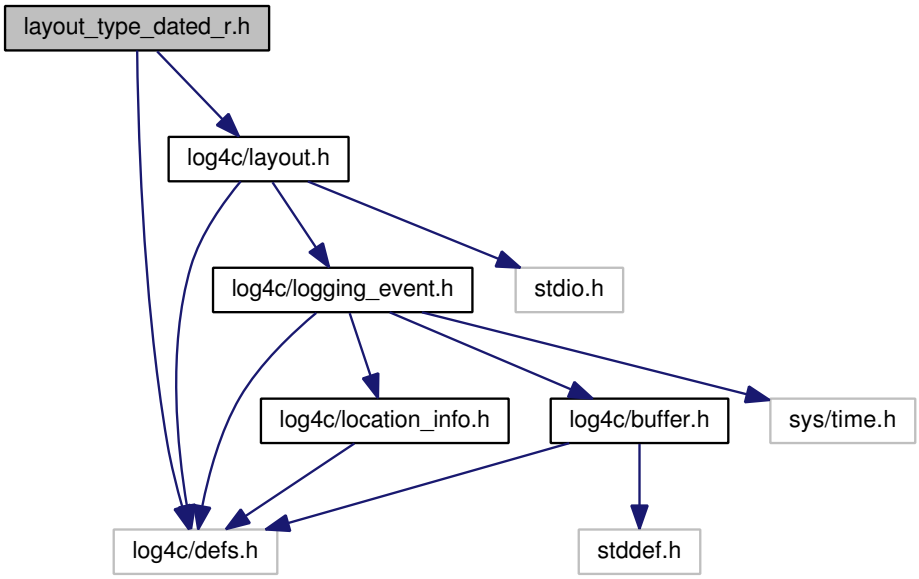
- "`%d`" is the date of the logging event
- "`%P`" is the priority of the logging event
- "`%c`" is the category of the logging event
- "`%m`" is the application supplied message associated with the logging event

8.16 layout_type_dated_r.h File Reference

Implement a dated_r layout.

```
#include <log4c/defs.h>
#include <log4c/layout.h>
```

Include dependency graph for layout_type_dated_r.h:



8.16.1 Detailed Description

Implement a dated_r layout.

In `log4j.PatternLayout` conventions, the dated_r layout has the following conversion pattern: `"%d %P %c - %m\n"`.

Where

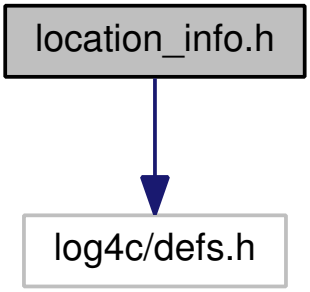
- `"%d"` is the date of the logging event
- `"%P"` is the priority of the logging event
- `"%c"` is the category of the logging event
- `"%m"` is the application supplied message associated with the logging event

8.17 location_info.h File Reference

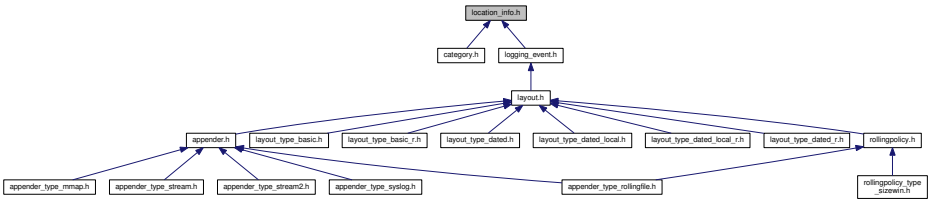
The internal representation of caller location information.

```
#include <log4c/defs.h>
```

Include dependency graph for location_info.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **log4c_location_info_t**

  logging location information

Macros

- #define **LOG4C_LOCATION_INFO_INITIALIZER**(user_data) { __FILE__, __LINE__, "(nil)", user_data }
- #define **log4c_location** __log4c_location(__LINE__)

8.17.1 Detailed Description

The internal representation of caller location information.

When a affirmative logging decision is made a **log4c_location_info_t** (p. 9) is created and is passed around the different log4c components.

8.17.2 Macro Definition Documentation

8.17.2.1 #define log4c_location __log4c_location(__LINE__)

This macro returns the literal representation of a logging event location

8.17.2.2 #define LOG4C_LOCATION_INFO_INITIALIZER(*user_data*) { __FILE__, __LINE__, "(nil)", user_data }

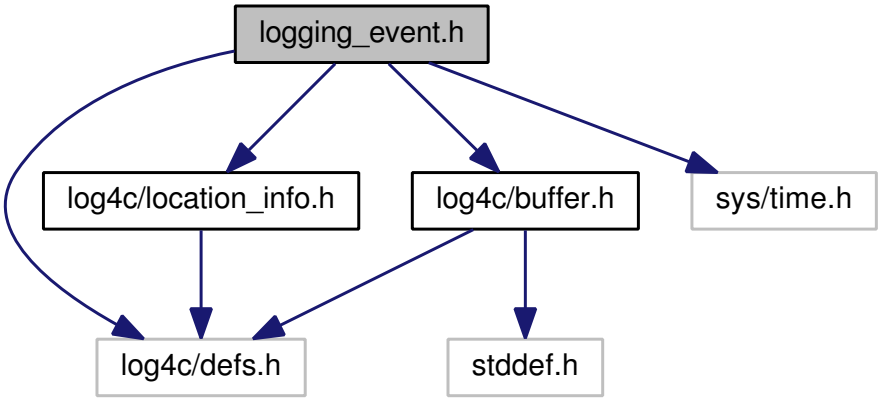
log4c_location_info_t (p. 9) initializer

8.18 logging_event.h File Reference

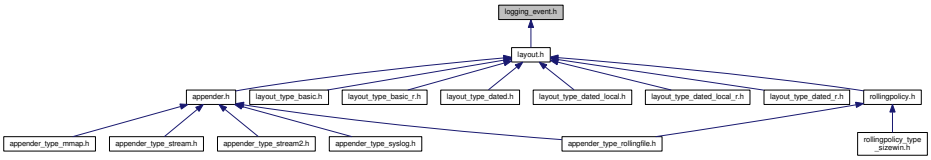
the internal representation of logging events.

```
#include <log4c/defs.h>
#include <log4c/buffer.h>
#include <log4c/location_info.h>
#include <sys/time.h>
```

Include dependency graph for logging_event.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **log4c_logging_event_t**
  logging event object

Functions

- **log4c_logging_event_t** ∗ **log4c_logging_event_new** (const char ∗a_category, int a_priority, const char ∗a_message)
- void **log4c_logging_event_delete** (**log4c_logging_event_t** ∗a_event)

8.18.1 Detailed Description

the internal representation of logging events.

When a affirmative logging decision is made a log4c_logging_event instance is created. This instance is passed around the different log4c components.

8.18.2 Function Documentation

8.18.2.1 void log4c_logging_event_delete (log4c_logging_event_t ∗ *a_event*)

Destructor for a logging event.

Parameters

a_event	the logging event object

8.18.2.2 log4c_logging_event_t* log4c_logging_event_new (const char * *a_category*, int *a_priority*, const char * *a_message*)

Constructor for a logging event.

Parameters

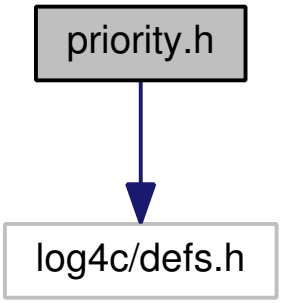
a_category	the category name
a_priority	the category initial priority
a_message	the message of this event

Todo need to handle multi-threading (NDC)

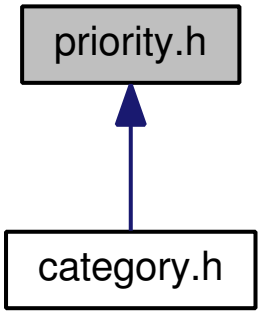
8.19 priority.h File Reference

The priority class provides importance levels with which one can categorize log messages.

`#include <log4c/defs.h>`
Include dependency graph for priority.h:



This graph shows which files directly or indirectly include this file:



Enumerations

Functions

- const char * **log4c_priority_to_string** (int a_priority)
- int **log4c_priority_to_int** (const char *a_priority_name)

8.19.1 Detailed Description

The priority class provides importance levels with which one can categorize log messages.

8.19.2 Enumeration Type Documentation

8.19.2.1 enum log4c_priority_level_t

Predefined Levels of priorities. These correspond to the priority levels used by syslog(3).

Enumerator

LOG4C_PRIORITY_FATAL fatal
LOG4C_PRIORITY_ALERT alert
LOG4C_PRIORITY_CRIT crit
LOG4C_PRIORITY_ERROR error
LOG4C_PRIORITY_WARN warn
LOG4C_PRIORITY_NOTICE notice
LOG4C_PRIORITY_INFO info
LOG4C_PRIORITY_DEBUG debug
LOG4C_PRIORITY_TRACE trace
LOG4C_PRIORITY_NOTSET notset
LOG4C_PRIORITY_UNKNOWN unknown

8.19.3 Function Documentation

8.19.3.1 int log4c_priority_to_int (const char * *a_priority_name*)

Parameters

a_priority_name	a priority string name.

Returns

the given numeric value of the priority.

References LOG4C_PRIORITY_UNKNOWN.

8.19.3.2 const char* log4c_priority_to_string (int *a_priority*)

Parameters

a_priority	a numeric value of the priority.

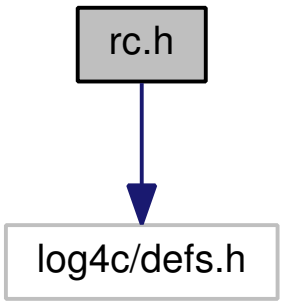
Returns

the given priority string name.

8.20 rc.h File Reference

log4c resource configuration

`#include <log4c/defs.h>`
Include dependency graph for rc.h:



Data Structures

- struct **log4c_rc_t**
  resource configuration object

Functions

- int **log4c_load** (const char *a_filename)

Variables

- **log4c_rc_t** *const **log4c_rc**

8.20.1 Detailed Description

log4c resource configuration

8.20.2 Function Documentation

8.20.2.1 int log4c_load (const char * *a_filename*)

load log4c resource configuration file

Parameters

a_filename	name of file to load

8.20.3 Variable Documentation

8.20.3.1 log4c_rc_t* const log4c_rc

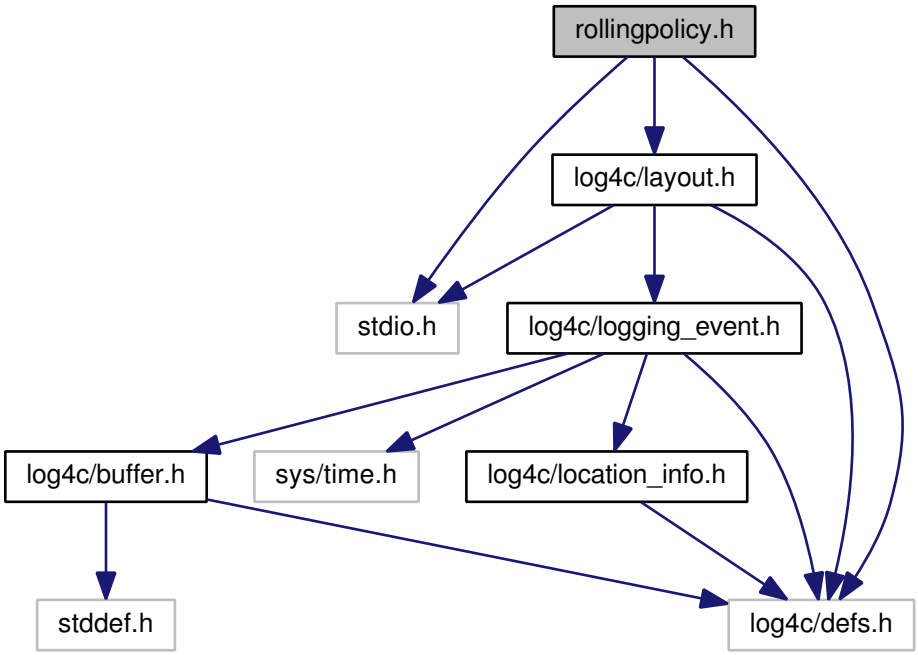
default log4c resource configuration object

8.21 rollingpolicy.h File Reference

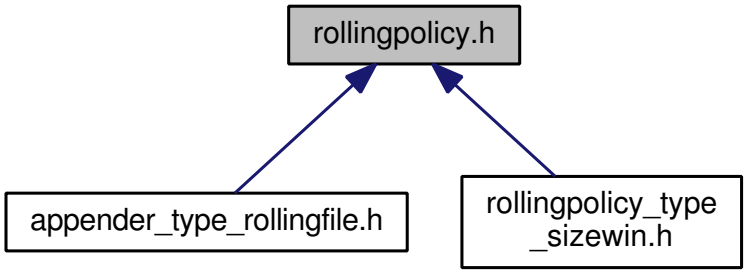
Log4c rolling policy interface. Defines the interface for managing and providing rolling policies.

```
#include <stdio.h>
#include <log4c/defs.h>
#include <log4c/layout.h>
```

Include dependency graph for rollingpolicy.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **log4c_rollingpolicy_type**

  log4c rollingpolicy type. Defines the interface a specific policy must provide to the rollingfile appender.

Macros

- #define **ROLLINGPOLICY_ROLLOVER_ERR_CAN_LOG** 0x05

Typedefs

- typedef struct __log4c_rollingpolicy **log4c_rollingpolicy_t**
- typedef struct **log4c_rollingpolicy_type** **log4c_rollingpolicy_type_t**

  log4c rollingpolicy type. Defines the interface a specific policy must provide to the rollingfile appender.

Functions

- **log4c_rollingpolicy_t** ∗ **log4c_rollingpolicy_get** (const char ∗policy_name)
- const **log4c_rollingpolicy_type_t** ∗ **log4c_rollingpolicy_type_set** (const **log4c_rollingpolicy_type_t** ∗a_type)
- void **log4c_rollingpolicy_set_udata** (**log4c_rollingpolicy_t** ∗policyp, void ∗udatap)
- int **log4c_rollingpolicy_init** (**log4c_rollingpolicy_t** ∗policyp, rollingfile_udata_t ∗rfup)
- int **log4c_rollingpolicy_fini** (**log4c_rollingpolicy_t** ∗policyp)
- int **log4c_rollingpolicy_is_triggering_event** (**log4c_rollingpolicy_t** ∗policyp, const **log4c_logging_event_t** ∗evtp, long current_file_size)
- const **log4c_rollingpolicy_type_t** ∗ **log4c_rollingpolicy_set_type** (**log4c_rollingpolicy_t** ∗a_rollingpolicy, const **log4c_rollingpolicy_type_t** ∗a_type)
- const **log4c_rollingpolicy_type_t** ∗ **log4c_rollingpolicy_type_get** (const char ∗a_name)
- void ∗ **log4c_rollingpolicy_get_udata** (const **log4c_rollingpolicy_t** ∗policyp)
- rollingfile_udata_t ∗ **log4c_rollingpolicy_get_rfudata** (const **log4c_rollingpolicy_t** ∗policyp)

8.21.1 Detailed Description

Log4c rolling policy interface. Defines the interface for managing and providing rolling policies.

A rolling policy is used to confogure a rollingfile appender to tell it when to trigger a rolover event.

8.21.2 Macro Definition Documentation

8.21.2.1 #define ROLLINGPOLICY_ROLLOVER_ERR_CAN_LOG 0x05

Effect a rollover according to policyp on the given file stream.

Parameters

policyp	pointer to the rolling policy
fp	filestream to rollover.

Returns

zero if successful, non-zero otherwise. The policy can return an indication that something went wrong but that the rollingfile appender can stull go ahead and log by returning an error code <= ROLLINGPOLICY_RO↩ LLOVER_ERR_CAN_LOG. Anything greater than means that the rolling file appender will not try to log it's message.

8.21.3 Typedef Documentation

8.21.3.1 typedef struct __log4c_rollingpolicy log4c_rollingpolicy_t

log4c rollingpolicy type

8.21.3.2 typedef struct log4c_rollingpolicy_type log4c_rollingpolicy_type_t

log4c rollingpolicy type. Defines the interface a specific policy must provide to the rollingfile appender.

Attributes description:

- `name` rollingpolicy type name
- `init()` init the rollingpolicy
- `is_triggering_event()`
- `rollover()`

8.21.4 Function Documentation

8.21.4.1 int log4c_rollingpolicy_fini (log4c_rollingpolicy_t * *policyp*)

Call the un initialization code of a rolling policy. This will call the fini routine of the particular rollingpolicy type to allow it to free up resources. If the call to fini in the rollingpolicy type fails then the rollingpolicy is not uninitialized. Try again later model...

Parameters

policyp	pointer to the rolling policy

Returns

zero if successful, non-zero otherwise.

8.21.4.2 log4c_rollingpolicy_t* log4c_rollingpolicy_get (const char * *policy_name*)

Get a new rolling policy

Parameters

policy_name	a name for the policy

Returns

a new rolling policy, otherwise NULL.

References log4c_rollingpolicy_fini().

8.21.4.3 rollingfile_udata_t* log4c_rollingpolicy_get_rfudata (const log4c_rollingpolicy_t * *policyp*)

Get the rollingfile appender associated with this policy.

Parameters

policyp	pointer to the rolling policy

Returns

pointer to the rolling file appender associated with this policy

8.21.4.4 void* log4c_rollingpolicy_get_udata (const log4c_rollingpolicy_t * *policyp*)

Get the rolling policy configuration.

Parameters

policyp	pointer to the rolling policy

Returns

pointer to the rolling policy configuration.

8.21.4.5 int log4c_rollingpolicy_init (log4c_rollingpolicy_t * *policyp,* rollingfile_udata_t * *rfup*)

Call the initialization code of a rolling policy.

Parameters

policyp	pointer to the rolling policy
rfup	the rolling appender user data this policy is used with

Returns

zero if successful, non-zero otherwise.

8.21.4.6 int log4c_rollingpolicy_is_triggering_event (log4c_rollingpolicy_t * *policyp,* const log4c_logging_event_t * *evtp,* long *current_file_size*)

Determine if a logging event should trigger a rollover according to the given policy.

Parameters

policyp	pointer to the rolling policy
evtp	the logging event pointer.
current_file_size	the size of the current file being logged to.

Returns

non-zero if rollover required, zero otherwise.

8.21.4.7 const log4c_rollingpolicy_type_t* log4c_rollingpolicy_set_type (log4c_rollingpolicy_t * *a_rollingpolicy,* const log4c_rollingpolicy_type_t * *a_type*)

sets the rolling policy type

Parameters

a_rollingpolicy	the log4c_rollingpolicy_t object
a_type	the new rollingpolicy type

Returns

the previous appender type

8.21.4.8 void log4c_rollingpolicy_set_udata (log4c_rollingpolicy_t * *policyp,* void * *udatap*)

Configure a rolling policy with a specific policy.

Parameters

policyp	pointer to the rolling policy
udatap	a specific policy type, for example sizewin.

Returns

zero if successful, non-zero otherwise.

8.21.4.9 const log4c_rollingpolicy_type_t* log4c_rollingpolicy_type_get (const char * *a_name*)

Get a pointer to an existing rollingpolicy type.

Parameters

a_name	the name of the rollingpolicy type to return.

Returns

a pointer to an existing rollingpolicy type, or NULL if no rollingpolicy type with the specified name exists.

8.21.4.10 const log4c_rollingpolicy_type_t* log4c_rollingpolicy_type_set (const log4c_rollingpolicy_type_t * *a_type*)

Use this function to register a rollingpolicy type with log4c. Once this is done you may refer to this type by name both programmatically and in the log4c configuration file.

Parameters

a_type	a pointer to the new rollingpolicy type to register.

Returns

a pointer to the previous rollingpolicy type of same name.

Example code fragment:

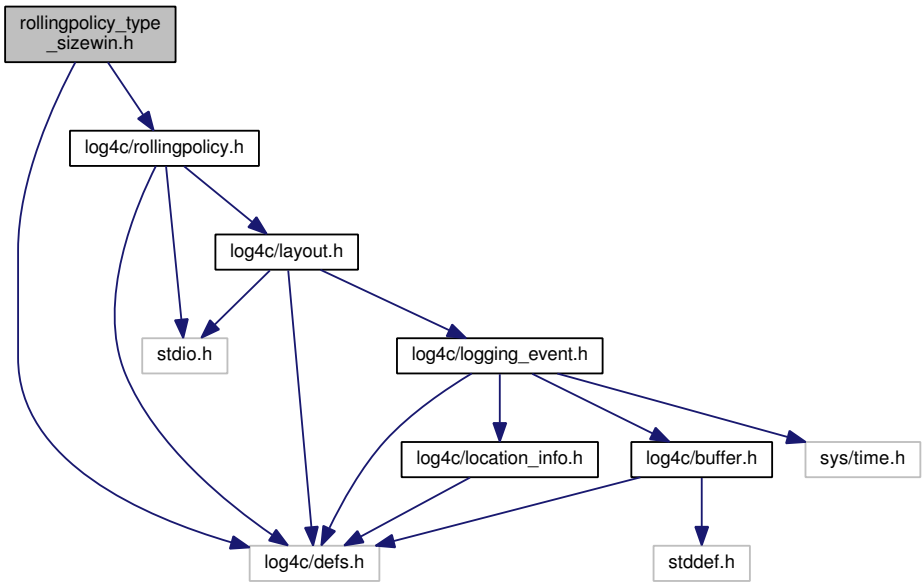
```
1 const log4c_rollingpolicy_type_t log4c_rollingpolicy_type_sizewin = {
2   "sizewin",
3   sizewin_init,
4   sizewin_is_triggering_event,
5   sizewin_rollover
6 };
7
8 log4c_rollingpolicy_type_set(&log4c_rollingpolicy_type_sizewin);
```

8.22 rollingpolicy_type_sizewin.h File Reference

Log4c rolling file size-win interface. Log4c ships with (and defaults to) the classic size-window rollover policy: this triggers rollover when files reach a maximum size. The first file in the list is always the current file; when a rollover event occurs files are shifted up by one position in the list–if the number of files in the list has already reached the max then the oldest file is rotated out of the window.

```
#include <log4c/defs.h>
#include <log4c/rollingpolicy.h>
```

Include dependency graph for rollingpolicy_type_sizewin.h:



Typedefs

- typedef struct __sizewin_udata **rollingpolicy_sizewin_udata_t**

Functions

- **rollingpolicy_sizewin_udata_t** * **sizewin_make_udata** (void)
- int **sizewin_udata_set_file_maxsize** (**rollingpolicy_sizewin_udata_t** *swup, long max_size)
- int **sizewin_udata_set_max_num_files** (**rollingpolicy_sizewin_udata_t** *swup, long max_num)
- int **sizewin_udata_set_appender** (**rollingpolicy_sizewin_udata_t** *swup, **log4c_appender_t** *app)

8.22.1 Detailed Description

Log4c rolling file size-win interface. Log4c ships with (and defaults to) the classic size-window rollover policy: this triggers rollover when files reach a maximum size. The first file in the list is always the current file; when a rollover event occurs files are shifted up by one position in the list–if the number of files in the list has already reached the max then the oldest file is rotated out of the window.

If the max file size is set to zero, this means 'no-limit'.

The default parameters for the size-win policy are 5 files of maximum size of 20kilobytes each. These parameters may be changed using the appropriate setter functions.

8.22.2 Typedef Documentation

8.22.2.1 typedef struct __sizewin_udata **rollingpolicy_sizewin_udata_t**

log4c size-win rolling policy type

8.22.3 Function Documentation

8.22.3.1 **rollingpolicy_sizewin_udata_t** * sizewin_make_udata (void)

Get a new size-win rolling policy

Returns

a new size-win rolling policy, otherwise NULL.

References sizewin_udata_set_file_maxsize(), and sizewin_udata_set_max_num_files().

8.22.3.2 int sizewin_udata_set_appender (**rollingpolicy_sizewin_udata_t** * *swup*, **log4c_appender_t** * *app*)

Set the rolling file appender in this rolling policy configuration.

Parameters

swup	the size-win configuration object.
app	the rolling file appender to set.

Returns

zero if successful, non-zero otherwise.

8.22.3.3 int sizewin_udata_set_file_maxsize (rollingpolicy_sizewin_udata_t * *swup*, long *max_size*)

Set the maximum file size in this rolling policy configuration.

Parameters

swup	the size-win configuration object.
max_size	the approximate maximum size any logging file will attain. If you set zero then it means 'no-limit' and so only one file of unlimited size will be used for logging.

Returns

zero if successful, non-zero otherwise.

8.22.3.4 int sizewin_udata_set_max_num_files (rollingpolicy_sizewin_udata_t * *swup*, long *max_num*)

Set the maximum number of filesin this rolling policy configuration.

Parameters

swup	the size-win configuration object.
max_num	the maximum number of files in the list.

Returns

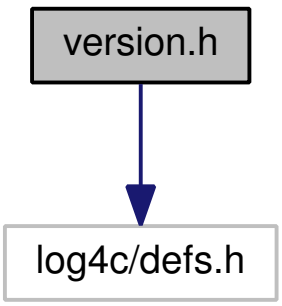
zero if successful, non-zero otherwise.

8.23 version.h File Reference

log4c version information

```
#include <log4c/defs.h>
```

Include dependency graph for version.h:



Macros

- #define **LOG4C_MAJOR_VERSION** 1
- #define **LOG4C_MINOR_VERSION** 2
- #define **LOG4C_MICRO_VERSION** 4

Functions

- const char * **log4c_version** (void)

Variables

- const int **log4c_major_version**
- const int **log4c_minor_version**
- const int **log4c_micro_version**

8.23.1 Detailed Description

log4c version information

8.23.2 Macro Definition Documentation

8.23.2.1 #define LOG4C_MAJOR_VERSION 1

constant macro holding the major version of log4c

8.23.2.2 #define LOG4C_MICRO_VERSION 4

constant macro holding the micro version of log4c

8.23.2.3 #define LOG4C_MINOR_VERSION 2

constant macro holding the minor version of log4c

8.23.3 Function Documentation

8.23.3.1 const char* log4c_version (void)

Returns

a string containing the full log4c version

8.23.4 Variable Documentation

8.23.4.1 const int log4c_major_version

constant variable holding the major version of log4c

8.23.4.2 const int log4c_micro_version

constant variable holding the micro version of log4c

8.23.4.3 const int log4c_minor_version

constant variable holding the minor version of log4c

Index

__log4c_category_trace
    category.h, 31

appender.h, 12
    log4c_appender_append, 14
    log4c_appender_close, 14
    log4c_appender_delete, 15
    log4c_appender_get, 15
    log4c_appender_get_layout, 15
    log4c_appender_get_name, 15
    log4c_appender_get_type, 15
    log4c_appender_get_udata, 16
    log4c_appender_new, 16
    log4c_appender_open, 16
    log4c_appender_print, 16
    log4c_appender_set_layout, 16
    log4c_appender_set_type, 17
    log4c_appender_set_udata, 17
    log4c_appender_t, 14
    log4c_appender_type_define, 13
    log4c_appender_type_get, 17
    log4c_appender_type_set, 17
    log4c_appender_type_t, 14
    log4c_appender_types_free, 18
    log4c_appender_types_print, 18
appender_type_mmap.h, 18
    log4c_appender_type_mmap, 19
appender_type_rollingfile.h, 20
    log4c_appender_type_rollingfile, 23
    rollingfile_get_current_file_size, 21
    rollingfile_make_udata, 21
    rollingfile_udata_get_files_prefix, 21
    rollingfile_udata_get_logdir, 22
    rollingfile_udata_set_files_prefix, 22
    rollingfile_udata_set_logdir, 22
    rollingfile_udata_set_policy, 22
appender_type_stream.h, 23
    log4c_appender_type_stream, 24
appender_type_stream2.h, 24
    log4c_appender_type_stream2, 27
    log4c_stream2_get_flags, 26
    log4c_stream2_get_fp, 26
    log4c_stream2_set_flags, 27
    log4c_stream2_set_fp, 27
appender_type_syslog.h, 27
    log4c_appender_type_syslog, 28

buffer.h, 29

category.h, 29
    __log4c_category_trace, 31
    log4c_category_alert, 32
    log4c_category_crit, 32
    log4c_category_debug, 32
    log4c_category_define, 31
    log4c_category_delete, 32
    log4c_category_error, 33
    log4c_category_fatal, 33
    log4c_category_get, 33
    log4c_category_get_additivity, 33
    log4c_category_get_appender, 35
    log4c_category_get_chainedpriority, 35
    log4c_category_get_name, 35
    log4c_category_get_priority, 35
    log4c_category_info, 36
    log4c_category_is_alert_enabled, 36
    log4c_category_is_crit_enabled, 36
    log4c_category_is_debug_enabled, 37
    log4c_category_is_error_enabled, 37
    log4c_category_is_fatal_enabled, 37
    log4c_category_is_info_enabled, 37
    log4c_category_is_notice_enabled, 38
    log4c_category_is_priority_enabled, 38
    log4c_category_is_trace_enabled, 38
    log4c_category_is_warn_enabled, 39
    log4c_category_list, 39
    log4c_category_log, 39
    log4c_category_log_locinfo, 40
    log4c_category_new, 40
    log4c_category_notice, 40
    log4c_category_print, 40
    log4c_category_set_additivity, 41
    log4c_category_set_appender, 41
    log4c_category_set_priority, 41
    log4c_category_t, 31
    log4c_category_warn, 42

init.h, 42
    log4c_fini, 43
    log4c_init, 43

LOG4C_LOCATION_INFO_INITIALIZER
    location_info.h, 56
LOG4C_MAJOR_VERSION
    version.h, 69
LOG4C_MICRO_VERSION
    version.h, 69
LOG4C_MINOR_VERSION
    version.h, 69
LOG4C_PRIORITY_ALERT
    priority.h, 59
LOG4C_PRIORITY_CRIT
    priority.h, 59
LOG4C_PRIORITY_DEBUG
    priority.h, 59
LOG4C_PRIORITY_ERROR
    priority.h, 59
LOG4C_PRIORITY_FATAL
    priority.h, 59
LOG4C_PRIORITY_INFO
    priority.h, 59
LOG4C_PRIORITY_NOTICE

priority.h, 59
LOG4C_PRIORITY_NOTSET
priority.h, 59
LOG4C_PRIORITY_TRACE
priority.h, 59
LOG4C_PRIORITY_UNKNOWN
priority.h, 59
LOG4C_PRIORITY_WARN
priority.h, 59
layout.h, 43
log4c_layout_delete, 45
log4c_layout_format, 45
log4c_layout_get, 45
log4c_layout_get_name, 46
log4c_layout_get_type, 46
log4c_layout_get_udata, 46
log4c_layout_new, 46
log4c_layout_print, 46
log4c_layout_set_type, 48
log4c_layout_set_udata, 48
log4c_layout_t, 45
log4c_layout_type_define, 44
log4c_layout_type_get, 48
log4c_layout_type_set, 48
log4c_layout_type_t, 45
log4c_layout_types_free, 49
log4c_layout_types_print, 49
layout_type_basic.h, 49
layout_type_basic_r.h, 50
layout_type_dated.h, 51
layout_type_dated_local.h, 52
layout_type_dated_local_r.h, 53
layout_type_dated_r.h, 54
location_info.h, 55
LOG4C_LOCATION_INFO_INITIALIZER, 56
log4c_location, 56
log4c_appender_append
appender.h, 14
log4c_appender_close
appender.h, 14
log4c_appender_delete
appender.h, 15
log4c_appender_get
appender.h, 15
log4c_appender_get_layout
appender.h, 15
log4c_appender_get_name
appender.h, 15
log4c_appender_get_type
appender.h, 15
log4c_appender_get_udata
appender.h, 16
log4c_appender_new
appender.h, 16
log4c_appender_open
appender.h, 16
log4c_appender_print
appender.h, 16
log4c_appender_set_layout
appender.h, 16
log4c_appender_set_type
appender.h, 17
log4c_appender_set_udata
appender.h, 17
log4c_appender_t
appender.h, 14
log4c_appender_type, 8
log4c_appender_type_define
appender.h, 13
log4c_appender_type_get
appender.h, 17
log4c_appender_type_mmap
appender_type_mmap.h, 19
log4c_appender_type_rollingfile
appender_type_rollingfile.h, 23
log4c_appender_type_set
appender.h, 17
log4c_appender_type_stream
appender_type_stream.h, 24
log4c_appender_type_stream2
appender_type_stream2.h, 27
log4c_appender_type_syslog
appender_type_syslog.h, 28
log4c_appender_type_t
appender.h, 14
log4c_appender_types_free
appender.h, 18
log4c_appender_types_print
appender.h, 18
log4c_buffer_t, 8
log4c_category_alert
category.h, 32
log4c_category_crit
category.h, 32
log4c_category_debug
category.h, 32
log4c_category_define
category.h, 31
log4c_category_delete
category.h, 32
log4c_category_error
category.h, 33
log4c_category_fatal
category.h, 33
log4c_category_get
category.h, 33
log4c_category_get_additivity
category.h, 33
log4c_category_get_appender
category.h, 35
log4c_category_get_chainedpriority
category.h, 35
log4c_category_get_name
category.h, 35
log4c_category_get_priority
category.h, 35

log4c_category_info
category.h, 36
log4c_category_is_alert_enabled
category.h, 36
log4c_category_is_crit_enabled
category.h, 36
log4c_category_is_debug_enabled
category.h, 37
log4c_category_is_error_enabled
category.h, 37
log4c_category_is_fatal_enabled
category.h, 37
log4c_category_is_info_enabled
category.h, 37
log4c_category_is_notice_enabled
category.h, 38
log4c_category_is_priority_enabled
category.h, 38
log4c_category_is_trace_enabled
category.h, 38
log4c_category_is_warn_enabled
category.h, 39
log4c_category_list
category.h, 39
log4c_category_log
category.h, 39
log4c_category_log_locinfo
category.h, 40
log4c_category_new
category.h, 40
log4c_category_notice
category.h, 40
log4c_category_print
category.h, 40
log4c_category_set_additivity
category.h, 41
log4c_category_set_appender
category.h, 41
log4c_category_set_priority
category.h, 41
log4c_category_t
category.h, 31
log4c_category_warn
category.h, 42
log4c_fini
init.h, 43
log4c_init
init.h, 43
log4c_layout_delete
layout.h, 45
log4c_layout_format
layout.h, 45
log4c_layout_get
layout.h, 45
log4c_layout_get_name
layout.h, 46
log4c_layout_get_type
layout.h, 46
log4c_layout_get_udata
layout.h, 46
log4c_layout_new
layout.h, 46
log4c_layout_print
layout.h, 46
log4c_layout_set_type
layout.h, 48
log4c_layout_set_udata
layout.h, 48
log4c_layout_t
layout.h, 45
log4c_layout_type, 9
log4c_layout_type_define
layout.h, 44
log4c_layout_type_get
layout.h, 48
log4c_layout_type_set
layout.h, 48
log4c_layout_type_t
layout.h, 45
log4c_layout_types_free
layout.h, 49
log4c_layout_types_print
layout.h, 49
log4c_load
rc.h, 60
log4c_location
location_info.h, 56
log4c_location_info_t, 9
log4c_logging_event_delete
logging_event.h, 57
log4c_logging_event_new
logging_event.h, 58
log4c_logging_event_t, 10
log4c_major_version
version.h, 69
log4c_micro_version
version.h, 69
log4c_minor_version
version.h, 69
log4c_priority_level_t
priority.h, 59
log4c_priority_to_int
priority.h, 59
log4c_priority_to_string
priority.h, 59
log4c_rc
rc.h, 61
log4c_rc_t, 11
log4c_rollingpolicy_fini
rollingpolicy.h, 63
log4c_rollingpolicy_get
rollingpolicy.h, 63
log4c_rollingpolicy_get_rfudata
rollingpolicy.h, 64
log4c_rollingpolicy_get_udata
rollingpolicy.h, 64

log4c_rollingpolicy_init
rollingpolicy.h, 64
log4c_rollingpolicy_is_triggering_event
rollingpolicy.h, 64
log4c_rollingpolicy_set_type
rollingpolicy.h, 65
log4c_rollingpolicy_set_udata
rollingpolicy.h, 65
log4c_rollingpolicy_t
rollingpolicy.h, 63
log4c_rollingpolicy_type, 11
log4c_rollingpolicy_type_get
rollingpolicy.h, 65
log4c_rollingpolicy_type_set
rollingpolicy.h, 66
log4c_rollingpolicy_type_t
rollingpolicy.h, 63
log4c_stream2_get_flags
appender_type_stream2.h, 26
log4c_stream2_get_fp
appender_type_stream2.h, 26
log4c_stream2_set_flags
appender_type_stream2.h, 27
log4c_stream2_set_fp
appender_type_stream2.h, 27
log4c_version
version.h, 69
logging_event.h, 57
log4c_logging_event_delete, 57
log4c_logging_event_new, 58

priority.h, 58
LOG4C_PRIORITY_ALERT, 59
LOG4C_PRIORITY_CRIT, 59
LOG4C_PRIORITY_DEBUG, 59
LOG4C_PRIORITY_ERROR, 59
LOG4C_PRIORITY_FATAL, 59
LOG4C_PRIORITY_INFO, 59
LOG4C_PRIORITY_NOTICE, 59
LOG4C_PRIORITY_NOTSET, 59
LOG4C_PRIORITY_TRACE, 59
LOG4C_PRIORITY_UNKNOWN, 59
LOG4C_PRIORITY_WARN, 59
log4c_priority_level_t, 59
log4c_priority_to_int, 59
log4c_priority_to_string, 59

ROLLINGPOLICY_ROLLOVER_ERR_CAN_LOG
rollingpolicy.h, 62
rc.h, 60
log4c_load, 60
log4c_rc, 61
rollingfile_get_current_file_size
appender_type_rollingfile.h, 21
rollingfile_make_udata
appender_type_rollingfile.h, 21
rollingfile_udata_get_files_prefix
appender_type_rollingfile.h, 21
rollingfile_udata_get_logdir
appender_type_rollingfile.h, 22
rollingfile_udata_set_files_prefix
appender_type_rollingfile.h, 22
rollingfile_udata_set_logdir
appender_type_rollingfile.h, 22
rollingfile_udata_set_policy
appender_type_rollingfile.h, 22
rollingpolicy.h, 61
log4c_rollingpolicy_fini, 63
log4c_rollingpolicy_get, 63
log4c_rollingpolicy_get_rfudata, 64
log4c_rollingpolicy_get_udata, 64
log4c_rollingpolicy_init, 64
log4c_rollingpolicy_is_triggering_event, 64
log4c_rollingpolicy_set_type, 65
log4c_rollingpolicy_set_udata, 65
log4c_rollingpolicy_t, 63
log4c_rollingpolicy_type_get, 65
log4c_rollingpolicy_type_set, 66
log4c_rollingpolicy_type_t, 63
ROLLINGPOLICY_ROLLOVER_ERR_CAN_LOG, 62
rollingpolicy_sizewin_udata_t
rollingpolicy_type_sizewin.h, 67
rollingpolicy_type_sizewin.h, 66
rollingpolicy_sizewin_udata_t, 67
sizewin_make_udata, 67
sizewin_udata_set_appender, 67
sizewin_udata_set_file_maxsize, 68
sizewin_udata_set_max_num_files, 68

sizewin_make_udata
rollingpolicy_type_sizewin.h, 67
sizewin_udata_set_appender
rollingpolicy_type_sizewin.h, 67
sizewin_udata_set_file_maxsize
rollingpolicy_type_sizewin.h, 68
sizewin_udata_set_max_num_files
rollingpolicy_type_sizewin.h, 68

version.h, 68
LOG4C_MAJOR_VERSION, 69
LOG4C_MICRO_VERSION, 69
LOG4C_MINOR_VERSION, 69
log4c_major_version, 69
log4c_micro_version, 69
log4c_minor_version, 69
log4c_version, 69