

log4c  
1.2.4

Generated by Doxygen 1.8.6

Fri May 9 2014 01:09:04

## Contents

<b>1</b>	<b>Log4c : Logging for C Library</b>	<b>2</b>
1.1	Summary . . . . .	2
1.2	Requirements . . . . .	2
1.2.1	Platforms . . . . .	2
1.2.2	Software . . . . .	2
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 . . . . .	3
1.4.3	Environment variables . . . . .	4
1.5	Customization . . . . .	4
1.6	Thanks . . . . .	4
1.7	Copyright . . . . .	4
<b>2</b>	<b>Todo List</b>	<b>5</b>
<b>3</b>	<b>Deprecated List</b>	<b>5</b>
<b>4</b>	<b>Bug List</b>	<b>5</b>
<b>5</b>	<b>Data Structure Index</b>	<b>5</b>
5.1	Data Structures . . . . .	5
<b>6</b>	<b>File Index</b>	<b>6</b>
6.1	File List . . . . .	6
<b>7</b>	<b>Data Structure Documentation</b>	<b>7</b>
7.1	log4c_appender_type Struct Reference . . . . .	7
7.1.1	Detailed Description . . . . .	7
7.2	log4c_buffer_t Struct Reference . . . . .	8
7.2.1	Detailed Description . . . . .	8
7.3	log4c_layout_type Struct Reference . . . . .	8
7.3.1	Detailed Description . . . . .	8
7.4	log4c_location_info_t Struct Reference . . . . .	8
7.4.1	Detailed Description . . . . .	9
7.5	log4c_logging_event_t Struct Reference . . . . .	9
7.5.1	Detailed Description . . . . .	9
7.6	log4c_rc_t Struct Reference . . . . .	10
7.6.1	Detailed Description . . . . .	10
7.7	log4c_rollingpolicy_type Struct Reference . . . . .	10

7.7.1	Detailed Description . . . . .	10
<b>8</b>	<b>File Documentation</b>	<b>10</b>
8.1	appender.h File Reference . . . . .	10
8.1.1	Detailed Description . . . . .	12
8.1.2	Macro Definition Documentation . . . . .	12
8.1.3	Typedef Documentation . . . . .	12
8.1.4	Function Documentation . . . . .	13
8.2	appender_type_mmap.h File Reference . . . . .	17
8.2.1	Detailed Description . . . . .	18
8.2.2	Variable Documentation . . . . .	18
8.3	appender_type_rollingfile.h File Reference . . . . .	18
8.3.1	Detailed Description . . . . .	19
8.3.2	Function Documentation . . . . .	20
8.3.3	Variable Documentation . . . . .	21
8.4	appender_type_stream.h File Reference . . . . .	21
8.4.1	Detailed Description . . . . .	22
8.4.2	Variable Documentation . . . . .	22
8.5	appender_type_stream2.h File Reference . . . . .	23
8.5.1	Detailed Description . . . . .	23
8.5.2	Function Documentation . . . . .	24
8.5.3	Variable Documentation . . . . .	25
8.6	appender_type_syslog.h File Reference . . . . .	25
8.6.1	Detailed Description . . . . .	26
8.6.2	Variable Documentation . . . . .	26
8.7	buffer.h File Reference . . . . .	26
8.7.1	Detailed Description . . . . .	27
8.8	category.h File Reference . . . . .	27
8.8.1	Detailed Description . . . . .	28
8.8.2	Macro Definition Documentation . . . . .	28
8.8.3	Typedef Documentation . . . . .	28
8.8.4	Function Documentation . . . . .	29
8.9	init.h File Reference . . . . .	37
8.9.1	Detailed Description . . . . .	37
8.9.2	Function Documentation . . . . .	37
8.10	layout.h File Reference . . . . .	38
8.10.1	Detailed Description . . . . .	39
8.10.2	Macro Definition Documentation . . . . .	39
8.10.3	Typedef Documentation . . . . .	40
8.10.4	Function Documentation . . . . .	40

8.11 layout_type_basic.h File Reference . . . . .	42
8.11.1 Detailed Description . . . . .	43
8.12 layout_type_basic_r.h File Reference . . . . .	43
8.12.1 Detailed Description . . . . .	44
8.13 layout_type_dated.h File Reference . . . . .	44
8.13.1 Detailed Description . . . . .	45
8.14 layout_type_dated_local.h File Reference . . . . .	45
8.14.1 Detailed Description . . . . .	46
8.15 layout_type_dated_local_r.h File Reference . . . . .	46
8.15.1 Detailed Description . . . . .	47
8.16 layout_type_dated_r.h File Reference . . . . .	47
8.16.1 Detailed Description . . . . .	48
8.17 location_info.h File Reference . . . . .	48
8.17.1 Detailed Description . . . . .	49
8.17.2 Macro Definition Documentation . . . . .	49
8.18 logging_event.h File Reference . . . . .	49
8.18.1 Detailed Description . . . . .	50
8.18.2 Function Documentation . . . . .	50
8.19 priority.h File Reference . . . . .	51
8.19.1 Detailed Description . . . . .	52
8.19.2 Enumeration Type Documentation . . . . .	52
8.19.3 Function Documentation . . . . .	52
8.20 rc.h File Reference . . . . .	53
8.20.1 Detailed Description . . . . .	53
8.20.2 Function Documentation . . . . .	53
8.20.3 Variable Documentation . . . . .	53
8.21 rollingpolicy.h File Reference . . . . .	54
8.21.1 Detailed Description . . . . .	55
8.21.2 Macro Definition Documentation . . . . .	55
8.21.3 Typedef Documentation . . . . .	55
8.21.4 Function Documentation . . . . .	56
8.22 rollingpolicy_type_sizewin.h File Reference . . . . .	58
8.22.1 Detailed Description . . . . .	59
8.22.2 Typedef Documentation . . . . .	59
8.22.3 Function Documentation . . . . .	59
8.23 version.h File Reference . . . . .	61
8.23.1 Detailed Description . . . . .	61
8.23.2 Macro Definition Documentation . . . . .	61
8.23.3 Function Documentation . . . . .	62
8.23.4 Variable Documentation . . . . .	62

## 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. 9) 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:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE log4c SYSTEM "">

<log4c>

    <config>
        <bufsize>0</bufsize>
        <debug level="0"/>
        <nocleanup>0</nocleanup>
    </config>

    <!-- root category ===== -->
    <category name="root" priority="notice"/>

    <!-- default appenders ===== -->
    <appender name="stdout" type="stream" layout="basic"/>
    <appender name="stderr" type="stream" layout="dated"/>
    <appender name="syslog" type="syslog" layout="basic"/>

    <!-- default layouts ===== -->
    <layout name="basic" type="basic"/>
    <layout name="dated" type="dated"/>
</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. 10)

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. 38)

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. 31) (`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. 36) (`log4c_category_t *a_category, struct __log4c_appender *a_appender`)

need multiple appenders per category

### Class `log4c_location_info_t` (p. 8)

this is not used

### Global `log4c_logging_event_new` (p. 50) (`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. 12) (`a_type`)

This macro, and the static initialization of appenders in general, is deprecated. Use rather the `log4c_appender_type_set()` (p. 16) function to initialize your appenders before calling `log4c_init()` (p. 38)

### Global `log4c_layout_type_define` (p. 39) (`a_type`)

This macro, and the static initialization of layouts in general, is deprecated. Use rather the `log4c_layout_type_set()` (p. 42) function to initialize your appenders before calling `log4c_init()` (p. 38)

## 4 Bug List

### Global `log4c_appender_append` (p. 13) (`log4c_appender_t *a_appender, log4c_logging_event_t (p. 9) *a_event`)

is this the right place to open an appender ?

### Global `log4c_category_get` (p. 31) (`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:

<b>log4c_appender_type</b> Log4c appender type class	7
<b>log4c_buffer_t</b> Buffer object	8



<b>log4c_layout_type</b>	
Log4c layout type class	8
<b>log4c_location_info_t</b>	
Logging location information	8
<b>log4c_logging_event_t</b>	
Logging event object	9
<b>log4c_rc_t</b>	
Resource configuration object	10
<b>log4c_rollingpolicy_type</b>	
Log4c rollingpolicy type. Defines the interface a specific policy must provide to the rollingfile appender	10

## 6 File Index

### 6.1 File List

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

<b>appender.h</b>	
Implement this interface for your own strategies for printing log statements	10
<b>appender_type_mmap.h</b>	
Log4c mmap(2) appender interface	17
<b>appender_type_rollingfile.h</b>	
Log4c rolling file appender interface	18
<b>appender_type_stream.h</b>	
Log4c stream appender interface	21
<b>appender_type_stream2.h</b>	
Log4c stream2 appender interface	23
<b>appender_type_syslog.h</b>	
Log4c syslog(3) appender interface	25
<b>buffer.h</b>	
Log4c buffer	26
<b>category.h</b>	
Central class in the log4c package	27
<b>config-win32.h</b>	??
<b>init.h</b>	
Log4c constructors and destructors	37
<b>layout.h</b>	
Interface for user specific layout format of log4c_logging_event events	38
<b>layout_type_basic.h</b>	
Implement a basic layout	42
<b>layout_type_basic_r.h</b>	
Implement a basic_r layout	43

<b>layout_type_dated.h</b>	
Implement a dated layout	44
<b>layout_type_dated_local.h</b>	
Implement a dated layout with local time	45
<b>layout_type_dated_local_r.h</b>	
Implement a dated layout (reentrant) with local time	46
<b>layout_type_dated_r.h</b>	
Implement a dated_r layout	47
<b>location_info.h</b>	
The internal representation of caller location information	48
<b>logging_event.h</b>	
Internal representation of logging events	49
<b>priority.h</b>	
The priority class provides importance levels with which one can categorize log messages	51
<b>rc.h</b>	
Log4c resource configuration	53
<b>rollingpolicy.h</b>	
Log4c rolling policy interface. Defines the interface for managing and providing rolling policies	54
<b>rollingpolicy_type_sizewin.h</b>	
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	58
<b>version.h</b>	
Log4c version information	61

## 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:

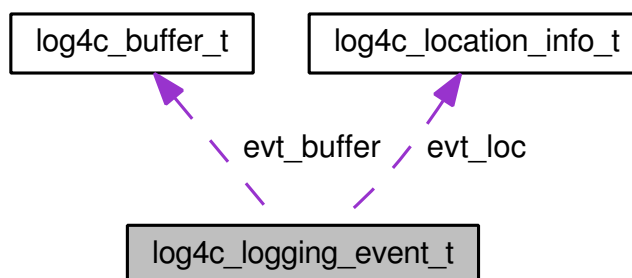
- **location\_info.h**

## 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. 37)
- `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.



- **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

<i>a_type</i>	the <b>log4c_appender_type_t</b> object to define
---------------	---

Warning

needs GCC support: otherwise this macro does nothing

**Deprecated** This macro, and the static initialization of appenders in general, is deprecated. Use rather the **log4c\_appender\_type\_set()** (p. 16) function to initialize your appenders before calling **log4c\_init()** (p. 38)

### 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

<i>a_appender</i>	the log4c_appender object
<i>a_event</i>	the <b>log4c_logging_event_t</b> (p. 9) 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

<i>a_appender</i>	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

<i>a_name</i>	the name of the appender to return.
---------------	-------------------------------------

###### Returns

a pointer to an existing appender, or NULL if no appender with the specified 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

<i>a_appender</i>	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

<i>a_appender</i>	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

<i>a_appender</i>	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

<i>a_appender</i>	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

<i>a_appender</i>	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

<i>a_appender</i>	the log4c_appender_t object
<i>a_stream</i>	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

<i>a_appender</i>	the log4c_appender_t object
<i>a_layout</i>	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

<i>a_appender</i>	the log4c_appender_t object
<i>a_type</i>	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

<i>a_appender</i>	the log4c_appender_t object
<i>a_udata</i>	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

<i>a_name</i>	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

<i>a_type</i>	a pointer to the new appender type to set.
---------------	--

## Returns

a pointer to the previous appender type of same name.

Example code fragment:

```
const log4c_appender_type_t log4c_appender_type_s13_file = {
    "s13_file",
    s13_file_open,
    s13_file_append,
    s13_file_close,
};

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

<i>fp</i>	the stream
-----------	------------

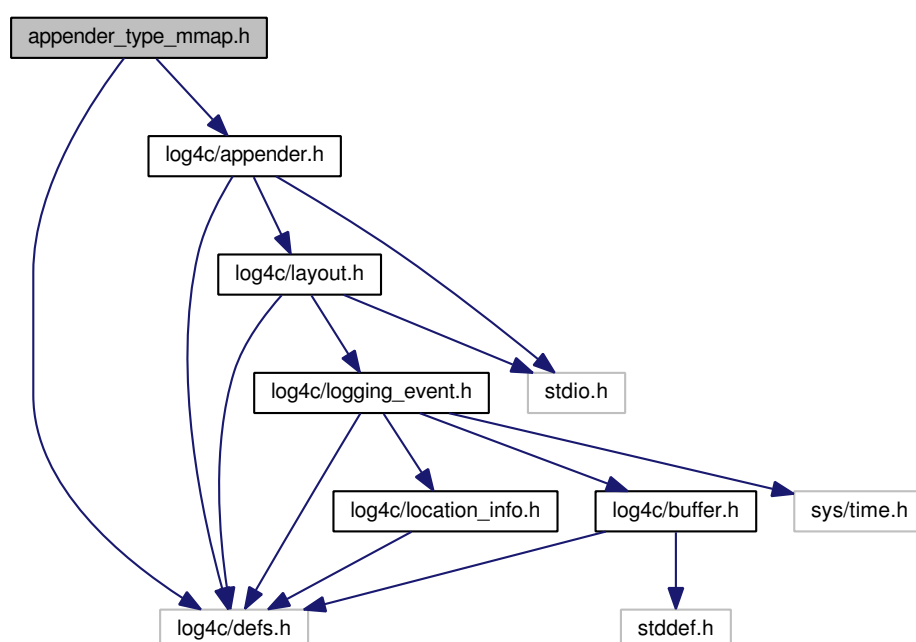
## 8.2 appender\_type\_mmap.h File Reference

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 mutiple 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.

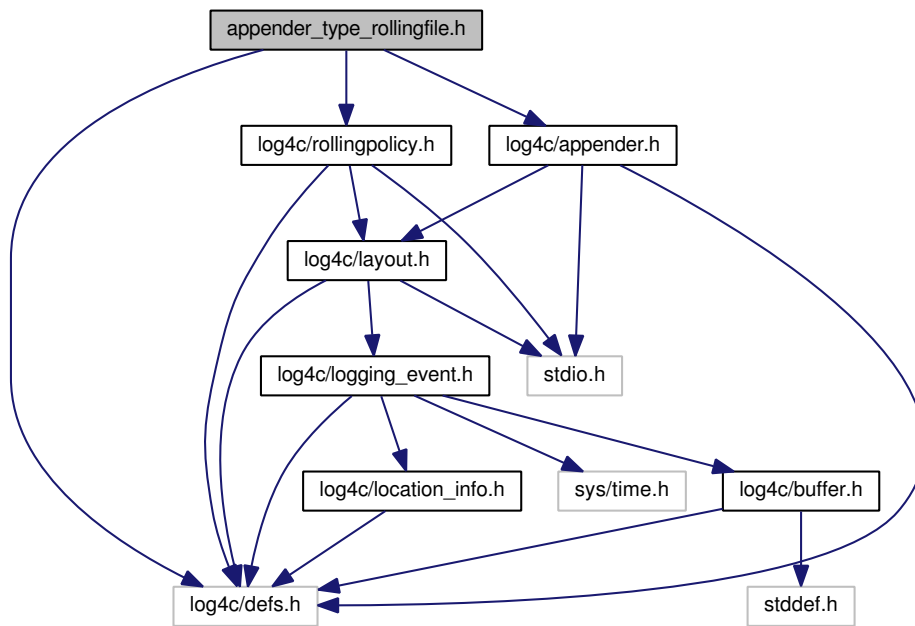
This should be used as a parameter to the `log4c_appender_set_type()` (p. 15) routine to set the type of the appender.

## 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 following 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.58) 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

<i>rfudatap</i>	the rolling file appender configuration object.
-----------------	---

##### Returns

the current size of the file being logged to.

#### 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

<i>rfudatap</i>	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

<i>rfudatap</i>	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

<i>rfudatap</i>	the rolling file appender configuration object.
<i>prefix</i>	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

<i>rfudatap</i>	the rolling file appender configuration object.
<i>logdir</i>	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

<i>rfudatap</i>	the rolling file appender configuration object.
<i>policyp</i>	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. 15) routine to set the type of the appender.

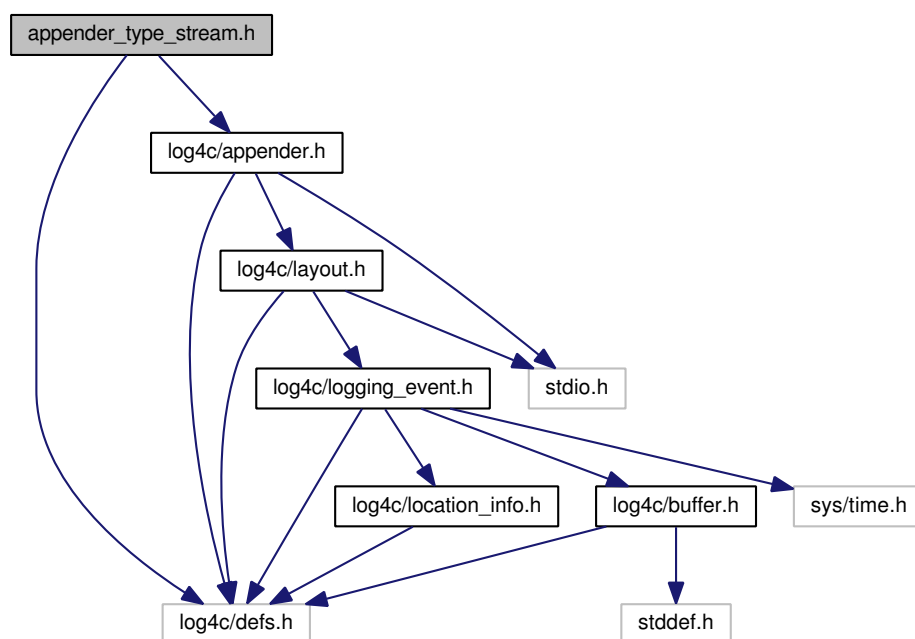
**8.4** appender\_type\_stream.h File Reference

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.15) 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. 15) routine to set the type of the appender.

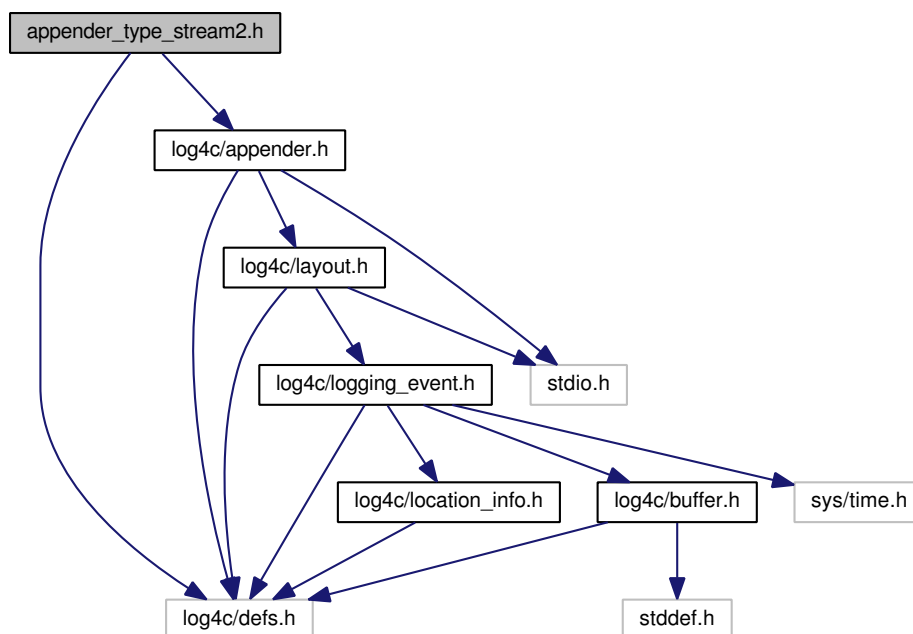
## 8.5 appender\_type\_stream2.h File Reference

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 `userdata` is not exposed directly. This means that new options (eg. configure the open mode ) could be added to stream2 while maintaining backward compatibility.

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

<i>this</i>	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

<i>this</i>	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

<i>this</i>	a pointer to the appender
<i>flags</i>	ar teh flags to set. These will overwrite the existing flags. Currently supported flags: LOG4-C_STREAM2_UNBUFFERED

## 8.5.2.4 void log4c\_stream2\_set\_fp ( log4c\_appender\_t \* a\_this, FILE \* fp )

Set the file pointer for this appender.

## Parameters

<i>this</i>	a pointer to the appender
<i>fp</i>	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. 15) routine to set the type of the appender.

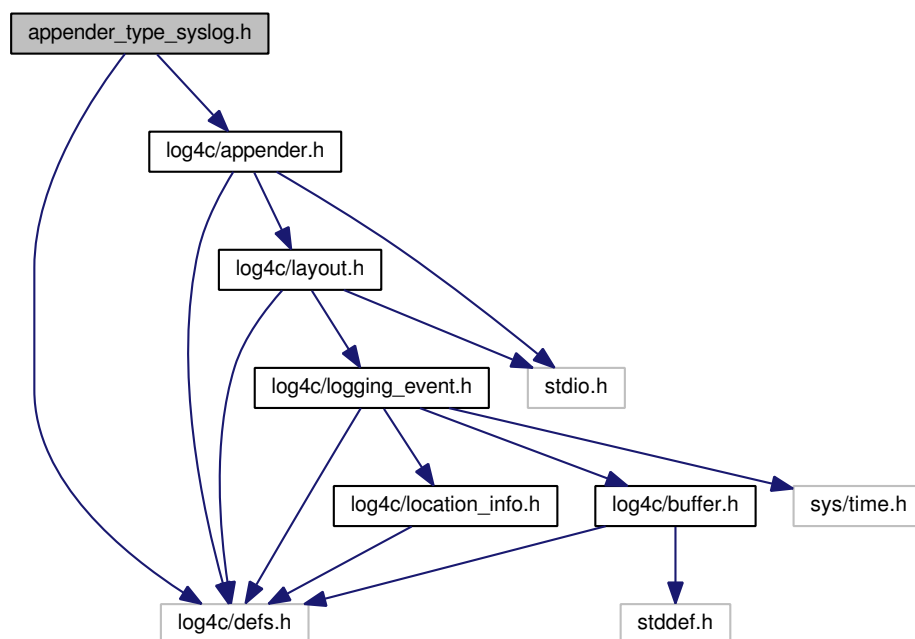
## 8.6 appender\_type\_syslog.h File Reference

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`

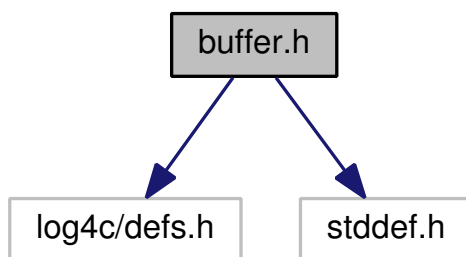
Syslog appender type definition.

This should be used as a parameter to the `log4c_appender_set_type()` (p. 15) routine to set the type of the appender.

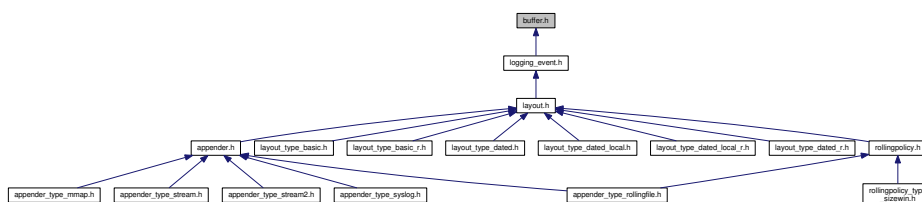
## 8.7 buffer.h File Reference

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

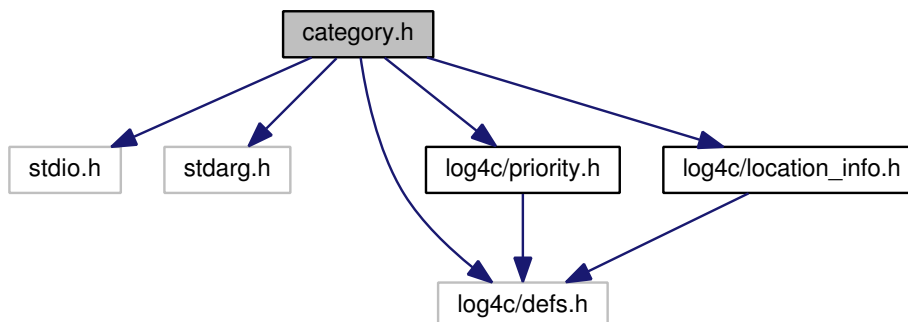
log4c buffer

## 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 distinctive 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

<i>a_category</i>	the log4c_category_t pointer name
<i>a_name</i>	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

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

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

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

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

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

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

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

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

<i>a_category</i>	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

<i>a_category</i>	the log4c_category_t object
<i>a_format</i>	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

<i>a_category</i>	the log4c_category_t object
<i>a_format</i>	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

<i>a_name</i>	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

<i>a_category</i>	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

<i>a_category</i>	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

<i>a_category</i>	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

<i>a_category</i>	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

<i>a_category</i>	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

<i>a_category</i>	the log4c_category_t object
<i>a_format</i>	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

<i>a_category</i>	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

<i>a_category</i>	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

<i>a_category</i>	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

<i>a_category</i>	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

<i>a_category</i>	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**

<i>a_category</i>	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**

<i>a_category</i>	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**

<i>a_category</i>	the log4c_category_t object
<i>a_priority</i>	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**

<i>a_category</i>	the log4c_category_t object
-------------------	-----------------------------

**Returns**

Whether the category will log.

References log4c\_category\_is\_priority\_enabled(), and LOG4C\_PRIORITY\_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

<i>a_category</i>	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

<i>a_cats</i>	array of categories that will be filled
<i>a_ncats</i>	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

<i>a_category</i>	the log4c_category_t object
<i>a_priority</i>	The priority of this log message.
<i>a_format</i>	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

<i>a_category</i>	the log4c_category_t object
<i>a_locinfo</i>	a user location info
<i>a_priority</i>	The priority of this log message.
<i>a_format</i>	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

<i>a_name</i>	the category name
---------------	-------------------

## Returns

a log4c\_category object

**Warning**

this method should not be called directly. You should use the **log4c\_category\_get()** (p. 31) 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**

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

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**

<i>a_category</i>	the log4c_category_t object
<i>a_stream</i>	The stream

References **log4c\_appender\_get\_name()**, **log4c\_category\_get\_name()**, and **log4c\_priority\_to\_string()**.

**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**

<i>a_category</i>	the log4c_category_t object
<i>a_additivity</i>	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**

<i>a_category</i>	the log4c_category_t object
<i>a_appender</i>	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

<i>a_category</i>	the log4c_category_t object
<i>a_priority</i>	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

<i>a_category</i>	the log4c_category_t object
<i>a_format</i>	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.

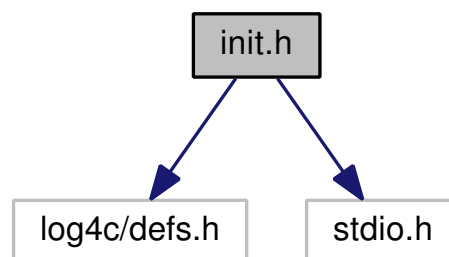
## 8.9 init.h File Reference

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_layout_types_free()`, and `log4c_rc`.

**8.9.2.2 int log4c\_init ( void )****constructor****Returns**

0 for success

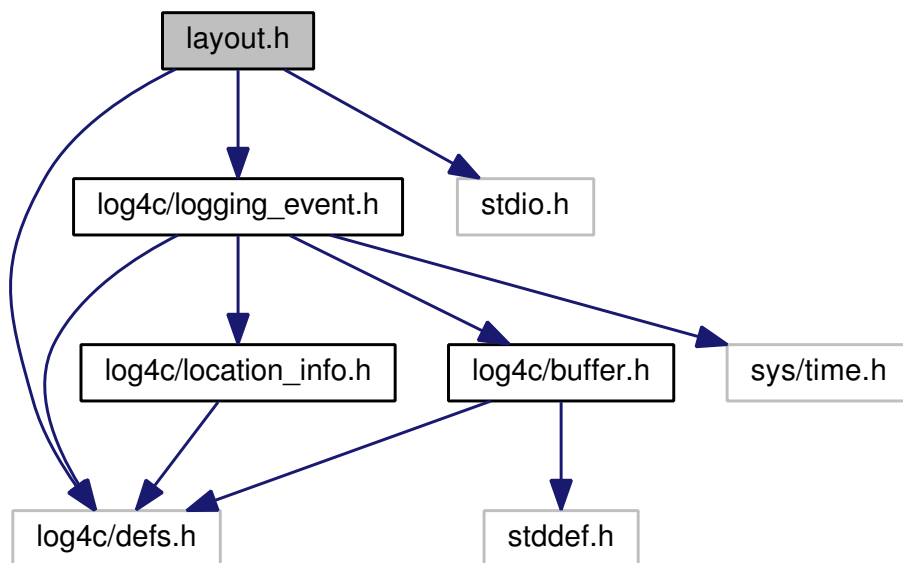
References `log4c_appender_get()`, `log4c_appender_type_set()`, `log4c_category_get()`, `log4c_category_set_appender()`, `log4c_category_set_priority()`, `log4c_layout_type_set()`, `log4c_load()`, `log4c_priority_to_int()`, and `log4c_rollingpolicy_type_set()`.

**8.10 layout.h File Reference**

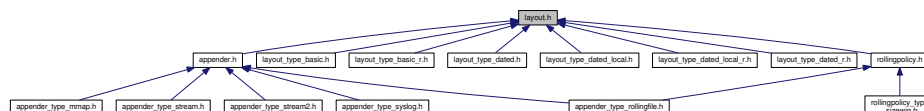
Interface for user specific layout format of `log4c_logging_event` events.

```
#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

<code>a_type</code>	the log4c_layout_type_t object to define
---------------------	--

## Warning

needs GCC support: otherwise this macro does nothing

**Deprecated** This macro, and the static initialization of layouts in general, is deprecated. Use rather the `log4c_layout_type_set()` (p. 42) function to initialize your appenders before calling `log4c_init()` (p. 38)

### 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

<i>a_layout</i>	the log4c_layout_t object
<i>a_event</i>	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

<i>a_name</i>	the name of the layout to return.
---------------	-----------------------------------

Returns

a pointer to an existing layout, or NULL if no layout with the specified 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

<i>a_layout</i>	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

<i>a_layout</i>	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

<i>a_layout</i>	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

<i>a_layout</i>	the log4c_layout_t object
<i>a_stream</i>	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

<i>a_layout</i>	the log4c_layout_t object
<i>a_type</i>	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

<i>a_layout</i>	the log4c_layout_t object
<i>a_udata</i>	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**

<i>a_name</i>	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**

<i>a_type</i>	a pointer to the new layout type to set.
---------------	--

**Returns**

a pointer to the previous layout type of same name.

Example code fragment:

```
const log4c_layout_type_t log4c_layout_type_xml = {
    "s13_xml",
    xml_format,
};

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**

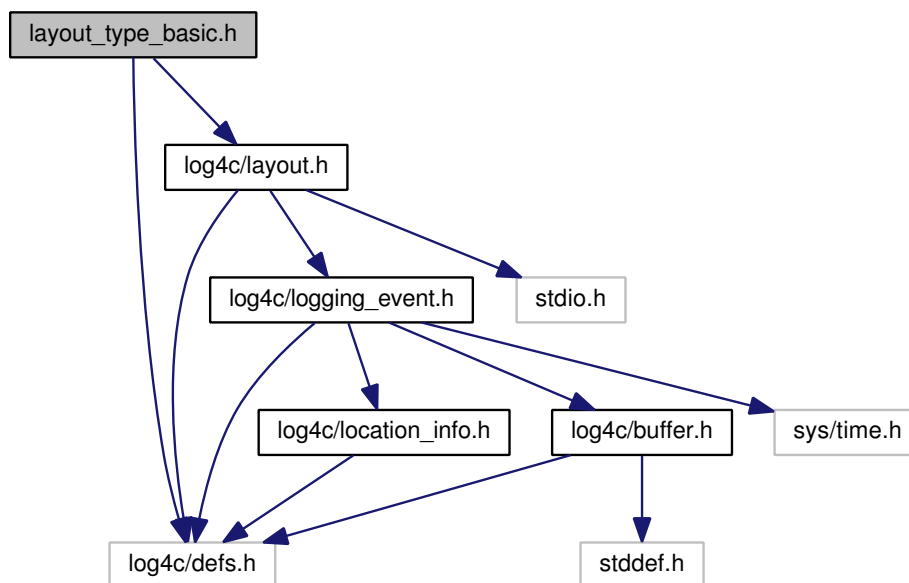
<i>fp</i>	the stream
-----------	------------

**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

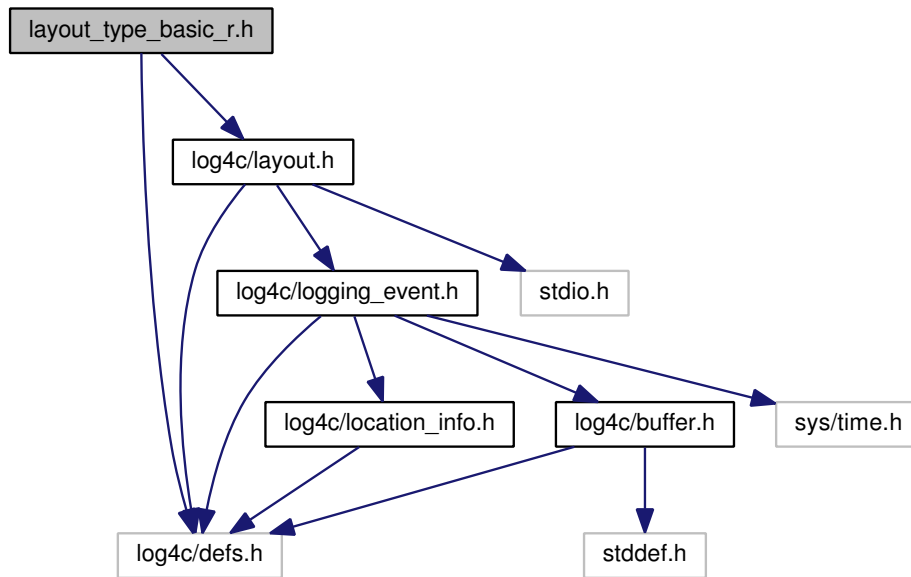
- `"%P"` is the priority of the logging event
- `"%C"` is the category of the logging event
- `"%m"` is the application supplied message associated with the logging event

## 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

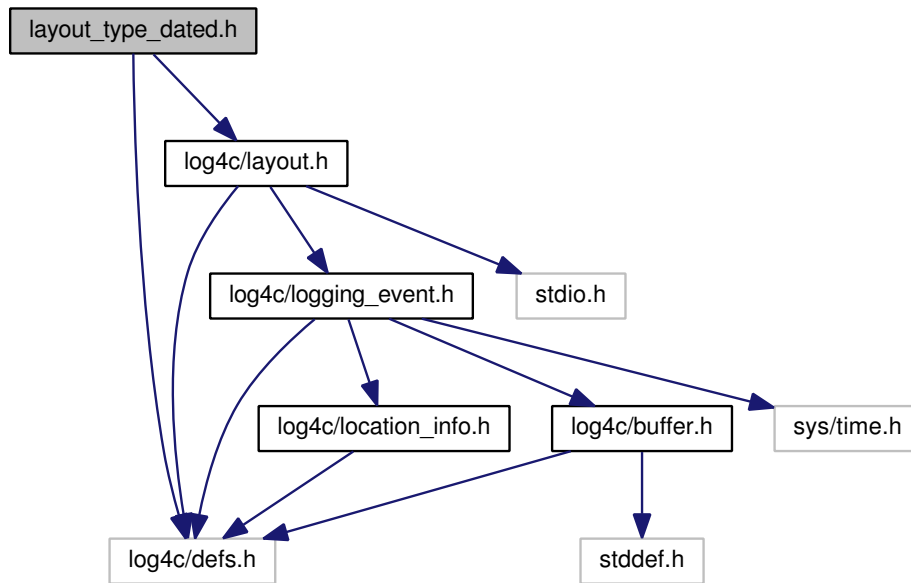
- `"%P"` is the priority of the logging event
- `"%C"` is the category of the logging event
- `"%m"` is the application supplied message associated with the logging event

### 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

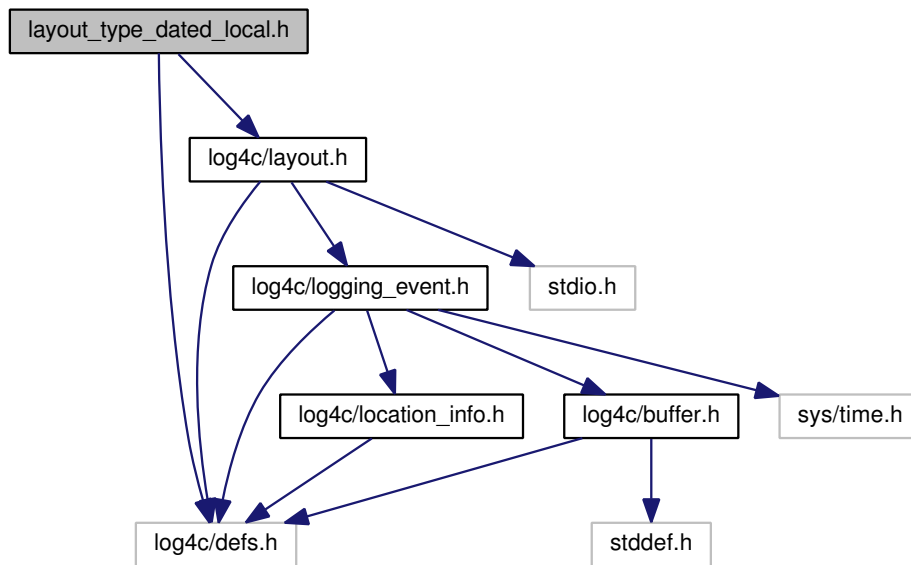
## 8.14 layout\_type\_dated\_local.h File Reference

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

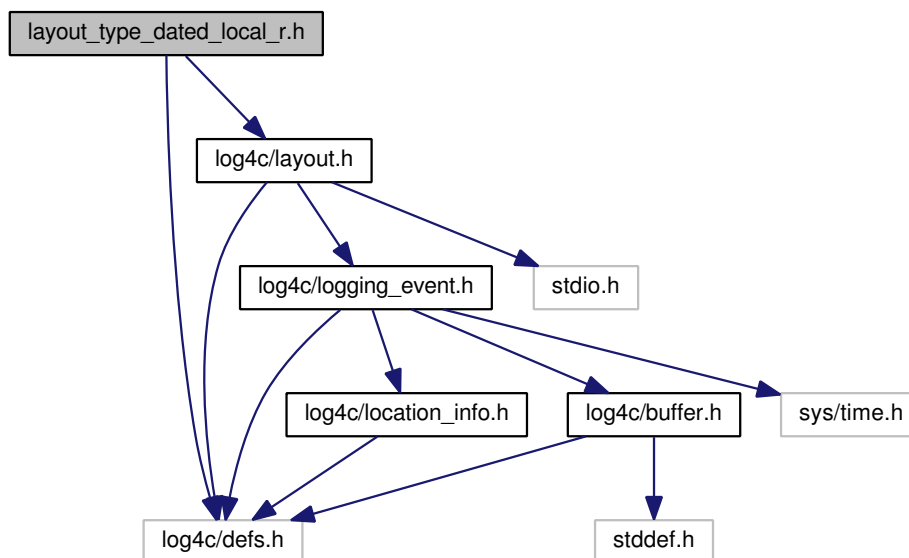
- `"%d"` is the date of the logging event
- `"%P"` is the priority of the logging event
- `"%C"` is the category of the logging event
- `"%m"` is the application supplied message associated with the logging event

#### 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

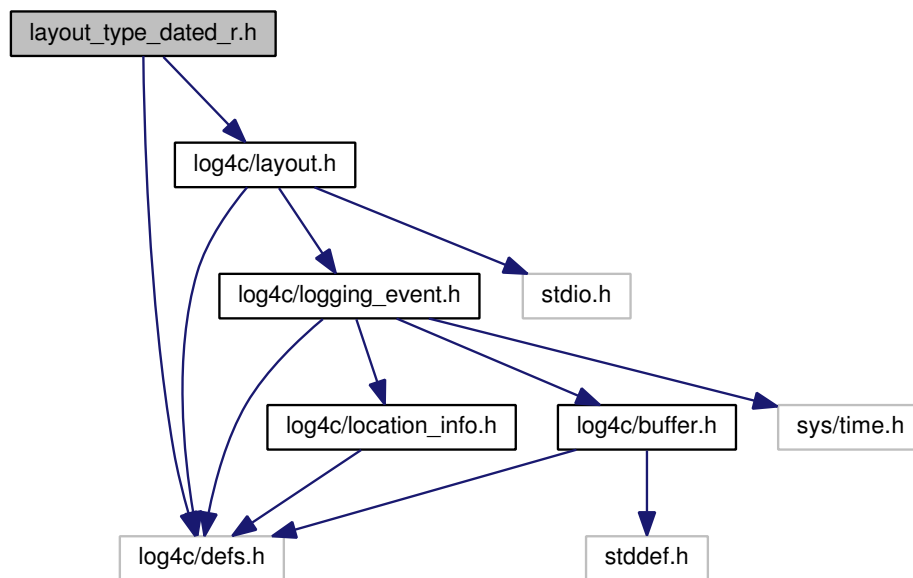
- `"%d"` is the date of the logging event
- `"%P"` is the priority of the logging event
- `"%C"` is the category of the logging event
- `"%m"` is the application supplied message associated with the logging event

## 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

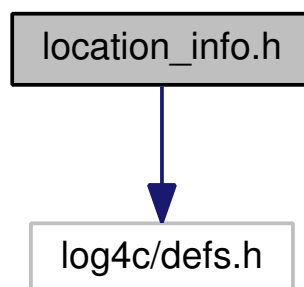
- `"%d"` is the date of the logging event
- `"%P"` is the priority of the logging event
- `"%c"` is the category of the logging event
- `"%m"` is the application supplied message associated with the logging event

## 8.17 location\_info.h File Reference

The internal representation of caller location information.

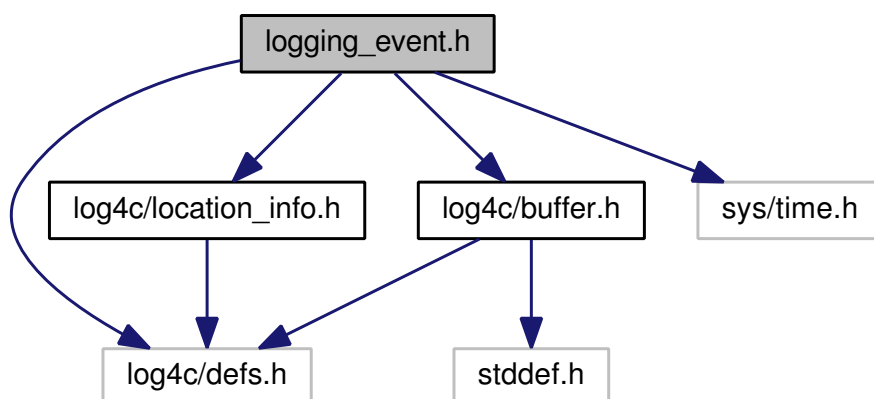
```
#include <log4c/defs.h>
```

Include dependency graph for `location_info.h`:

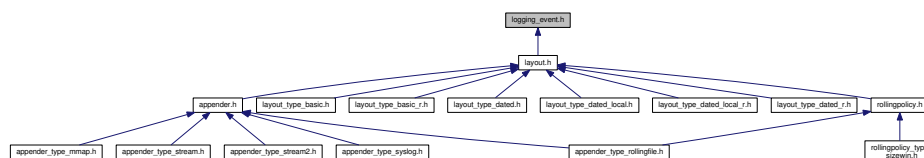




Include dependency graph for logging\_event.h:



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

<i>a_event</i>	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

<i>a_category</i>	the category name
<i>a_priority</i>	the category initial priority
<i>a_message</i>	the message of this event

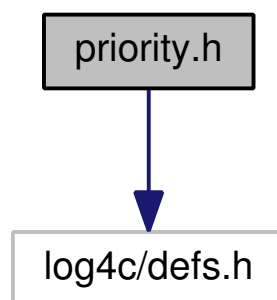
**Todo** need to handle multi-threading (NDC)

## 8.19 priority.h File Reference

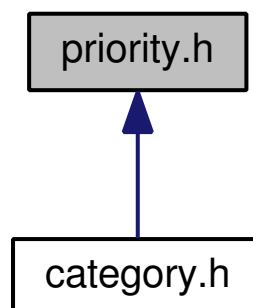
The priority class provides importance levels with which one can categorize log messages.

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

Include dependency graph for priority.h:



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



## Enumerations

- enum `log4c_priority_level_t`{  
    `LOG4C_PRIORITY_FATAL` = 000,  
    `LOG4C_PRIORITY_ALERT` = 100,  
    `LOG4C_PRIORITY_CRIT` = 200,  
    `LOG4C_PRIORITY_ERROR` = 300,  
    `LOG4C_PRIORITY_WARN` = 400,  
    `LOG4C_PRIORITY_NOTICE` = 500,  
    `LOG4C_PRIORITY_INFO` = 600,  
    `LOG4C_PRIORITY_DEBUG` = 700,  
    `LOG4C_PRIORITY_TRACE` = 800,  
    `LOG4C_PRIORITY_NOTSET` = 900,  
    `LOG4C_PRIORITY_UNKNOWN` = 1000 }

## 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

<i>a_priority_name</i>	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

<i>a_priority</i>	a numeric value of the priority.
-------------------	----------------------------------

##### Returns

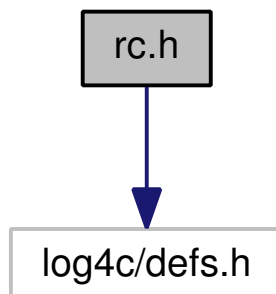
the given priority string name.

## 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

<i>a_filename</i>	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

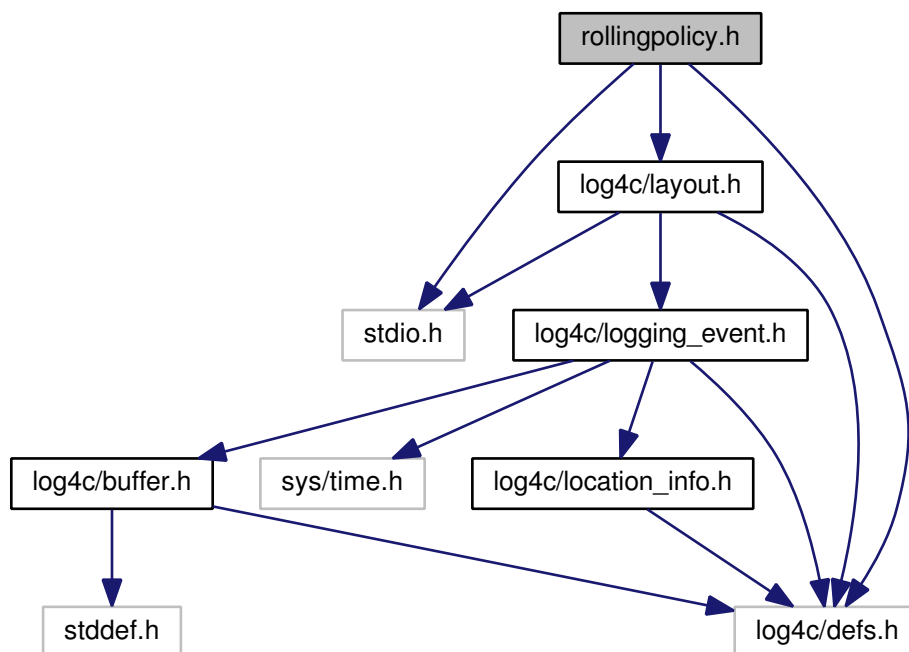
Log4c rolling policy interface. Defines the interface for managing and providing rolling policies.

```
#include <stdio.h>
```

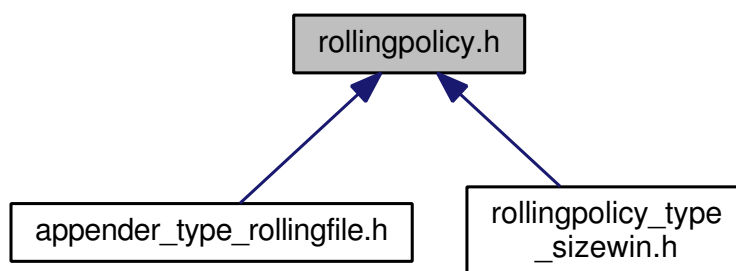
```
#include <log4c/defs.h>
```

```
#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 configure a rollingfile appender to tell it when to trigger a rollover 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

<i>policyp</i>	pointer to the rolling policy
<i>fp</i>	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 still go ahead and log by returning an error code  $\leq$  ROLLINGPOLICY\_ROLLOVER\_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\* *polycyp* )

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

<i>polycyp</i>	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

<i>policy_name</i>	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\* *polycyp* )

Get the rollingfile appender associated with this policy.

Parameters

<i>polycyp</i>	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\* *polycyp* )

Get the rolling policy configuration.

## Parameters

<i>polycyp</i>	pointer to the rolling policy
----------------	-------------------------------

## Returns

pointer to the rolling policy configuration.

8.21.4.5 `int log4c_rollingpolicy_init ( log4c_rollingpolicy_t * polycyp, rollingfile_udata_t * rfup )`

Call the initialization code of a rolling policy.

## Parameters

<i>polycyp</i>	pointer to the rolling policy
<i>rfup</i>	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 * polycyp, 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

<i>polycyp</i>	pointer to the rolling policy
<i>evtp</i>	the logging event pointer.
<i>current_file_size</i>	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

<i>a_rollingpolicy</i>	the log4c_rollingpolicy_t object
<i>a_type</i>	the new rollingpolicy type

## Returns

the previous appender type

8.21.4.8 `void log4c_rollingpolicy_set_udata ( log4c_rollingpolicy_t * polycyp, void * udatap )`

Configure a rolling policy with a specific policy.

## Parameters

<i>polycyp</i>	pointer to the rolling policy
<i>udatap</i>	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**

<i>a_name</i>	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**

<i>a_type</i>	a pointer to the new rollingpolicy type to register.
---------------	--

**Returns**

a pointer to the previous rollingpolicy type of same name.

Example code fragment:

```
const log4c_rollingpolicy_type_t log4c_rollingpolicy_type_sizewin = {
    "sizewin",
    sizewin_init,
    sizewin_is_triggering_event,
    sizewin_rollover
};

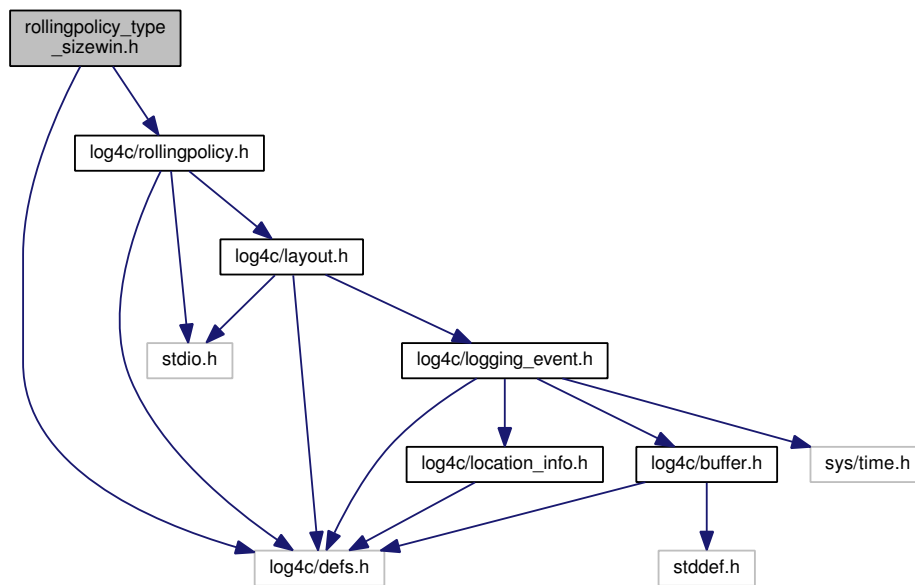
log4c_rollingpolicy_type_set (&log4c_rollingpolicy_type_sizewin);
```

**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**

<i>swup</i>	the size-win configuration object.
<i>app</i>	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**

<i>swup</i>	the size-win configuration object.
<i>max_size</i>	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 files in this rolling policy configuration.

**Parameters**

<i>swup</i>	the size-win configuration object.
<i>max_num</i>	the maximum number of files in the list.

#### Returns

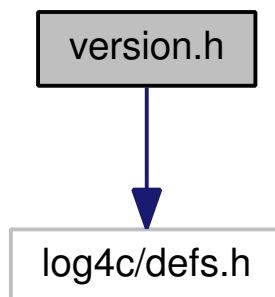
zero if successful, non-zero otherwise.

## 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, 29
- appender.h, 10
  - log4c\_appender\_append, 13
  - log4c\_appender\_close, 13
  - log4c\_appender\_delete, 13
  - log4c\_appender\_get, 13
  - log4c\_appender\_get\_layout, 13
  - log4c\_appender\_get\_name, 14
  - log4c\_appender\_get\_type, 14
  - log4c\_appender\_get\_udata, 14
  - log4c\_appender\_new, 14
  - log4c\_appender\_open, 14
  - log4c\_appender\_print, 14
  - log4c\_appender\_set\_layout, 15
  - log4c\_appender\_set\_type, 15
  - log4c\_appender\_set\_udata, 15
  - log4c\_appender\_t, 12
  - log4c\_appender\_type\_define, 12
  - log4c\_appender\_type\_get, 15
  - log4c\_appender\_type\_set, 15
  - log4c\_appender\_type\_t, 12
  - log4c\_appender\_types\_free, 17
  - log4c\_appender\_types\_print, 17
- appender\_type\_mmap.h, 17
  - log4c\_appender\_type\_mmap, 18
- appender\_type\_rollingfile.h, 18
  - log4c\_appender\_type\_rollingfile, 21
  - rollingfile\_get\_current\_file\_size, 20
  - rollingfile\_make\_udata, 20
  - rollingfile\_udata\_get\_files\_prefix, 20
  - rollingfile\_udata\_get\_logdir, 20
  - rollingfile\_udata\_set\_files\_prefix, 20
  - rollingfile\_udata\_set\_logdir, 21
  - rollingfile\_udata\_set\_policy, 21
- appender\_type\_stream.h, 21
  - log4c\_appender\_type\_stream, 22
- appender\_type\_stream2.h, 23
  - log4c\_appender\_type\_stream2, 25
  - log4c\_stream2\_get\_flags, 24
  - log4c\_stream2\_get\_fp, 24
  - log4c\_stream2\_set\_flags, 24
  - log4c\_stream2\_set\_fp, 25
- appender\_type\_syslog.h, 25
  - log4c\_appender\_type\_syslog, 26
- buffer.h, 26
- category.h, 27
  - \_\_log4c\_category\_trace, 29
  - log4c\_category\_alert, 29
  - log4c\_category\_crit, 29
  - log4c\_category\_debug, 29
  - log4c\_category\_define, 28
  - log4c\_category\_delete, 29
  - log4c\_category\_error, 29
  - log4c\_category\_fatal, 31
  - log4c\_category\_get, 31
  - log4c\_category\_get\_additivity, 31
  - log4c\_category\_get\_appender, 31
  - log4c\_category\_get\_chainedpriority, 31
  - log4c\_category\_get\_name, 32
  - log4c\_category\_get\_priority, 32
  - log4c\_category\_info, 32
  - log4c\_category\_is\_alert\_enabled, 32
  - log4c\_category\_is\_crit\_enabled, 32
  - log4c\_category\_is\_debug\_enabled, 33
  - log4c\_category\_is\_error\_enabled, 33
  - log4c\_category\_is\_fatal\_enabled, 33
  - log4c\_category\_is\_info\_enabled, 33
  - log4c\_category\_is\_notice\_enabled, 34
  - log4c\_category\_is\_priority\_enabled, 34
  - log4c\_category\_is\_trace\_enabled, 34
  - log4c\_category\_is\_warn\_enabled, 34
  - log4c\_category\_list, 35
  - log4c\_category\_log, 35
  - log4c\_category\_log\_locinfo, 35
  - log4c\_category\_new, 35
  - log4c\_category\_notice, 36
  - log4c\_category\_print, 36
  - log4c\_category\_set\_additivity, 36
  - log4c\_category\_set\_appender, 36
  - log4c\_category\_set\_priority, 36
  - log4c\_category\_t, 28
  - log4c\_category\_warn, 37
- init.h, 37
  - log4c\_fini, 37
  - log4c\_init, 38
- LOG4C\_LOCATION\_INFO\_INITIALIZER
  - location\_info.h, 49
- LOG4C\_MAJOR\_VERSION
  - version.h, 61
- LOG4C\_MICRO\_VERSION
  - version.h, 61
- LOG4C\_MINOR\_VERSION
  - version.h, 61
- LOG4C\_PRIORITY\_ALERT
  - priority.h, 52
- LOG4C\_PRIORITY\_CRIT
  - priority.h, 52
- LOG4C\_PRIORITY\_DEBUG
  - priority.h, 52
- LOG4C\_PRIORITY\_ERROR
  - priority.h, 52
- LOG4C\_PRIORITY\_FATAL
  - priority.h, 52
- LOG4C\_PRIORITY\_INFO
  - priority.h, 52
- LOG4C\_PRIORITY\_NOTICE

---

priority.h, 52  
 LOG4C\_PRIORITY\_NOTSET  
 priority.h, 52  
 LOG4C\_PRIORITY\_TRACE  
 priority.h, 52  
 LOG4C\_PRIORITY\_UNKNOWN  
 priority.h, 52  
 LOG4C\_PRIORITY\_WARN  
 priority.h, 52  
 layout.h, 38  
 log4c\_layout\_delete, 40  
 log4c\_layout\_format, 40  
 log4c\_layout\_get, 40  
 log4c\_layout\_get\_name, 40  
 log4c\_layout\_get\_type, 40  
 log4c\_layout\_get\_udata, 41  
 log4c\_layout\_new, 41  
 log4c\_layout\_print, 41  
 log4c\_layout\_set\_type, 41  
 log4c\_layout\_set\_udata, 41  
 log4c\_layout\_t, 40  
 log4c\_layout\_type\_define, 39  
 log4c\_layout\_type\_get, 41  
 log4c\_layout\_type\_set, 42  
 log4c\_layout\_type\_t, 40  
 log4c\_layout\_types\_free, 42  
 log4c\_layout\_types\_print, 42  
 layout\_type\_basic.h, 42  
 layout\_type\_basic\_r.h, 43  
 layout\_type\_dated.h, 44  
 layout\_type\_dated\_local.h, 45  
 layout\_type\_dated\_local\_r.h, 46  
 layout\_type\_dated\_r.h, 47  
 location\_info.h, 48  
 LOG4C\_LOCATION\_INFO\_INITIALIZER, 49  
 log4c\_location, 49  
 log4c\_appender\_append  
 appender.h, 13  
 log4c\_appender\_close  
 appender.h, 13  
 log4c\_appender\_delete  
 appender.h, 13  
 log4c\_appender\_get  
 appender.h, 13  
 log4c\_appender\_get\_layout  
 appender.h, 13  
 log4c\_appender\_get\_name  
 appender.h, 14  
 log4c\_appender\_get\_type  
 appender.h, 14  
 log4c\_appender\_get\_udata  
 appender.h, 14  
 log4c\_appender\_new  
 appender.h, 14  
 log4c\_appender\_open  
 appender.h, 14  
 log4c\_appender\_print  
 appender.h, 14  
 log4c\_appender\_set\_layout  
 appender.h, 15  
 log4c\_appender\_set\_type  
 appender.h, 15  
 log4c\_appender\_set\_udata  
 appender.h, 15  
 log4c\_appender\_t  
 appender.h, 12  
 log4c\_appender\_type, 7  
 log4c\_appender\_type\_define  
 appender.h, 12  
 log4c\_appender\_type\_get  
 appender.h, 15  
 log4c\_appender\_type\_mmap  
 appender\_type\_mmap.h, 18  
 log4c\_appender\_type\_rollingfile  
 appender\_type\_rollingfile.h, 21  
 log4c\_appender\_type\_set  
 appender.h, 15  
 log4c\_appender\_type\_stream  
 appender\_type\_stream.h, 22  
 log4c\_appender\_type\_stream2  
 appender\_type\_stream2.h, 25  
 log4c\_appender\_type\_syslog  
 appender\_type\_syslog.h, 26  
 log4c\_appender\_type\_t  
 appender.h, 12  
 log4c\_appender\_types\_free  
 appender.h, 17  
 log4c\_appender\_types\_print  
 appender.h, 17  
 log4c\_buffer\_t, 8  
 log4c\_category\_alert  
 category.h, 29  
 log4c\_category\_crit  
 category.h, 29  
 log4c\_category\_debug  
 category.h, 29  
 log4c\_category\_define  
 category.h, 28  
 log4c\_category\_delete  
 category.h, 29  
 log4c\_category\_error  
 category.h, 29  
 log4c\_category\_fatal  
 category.h, 31  
 log4c\_category\_get  
 category.h, 31  
 log4c\_category\_get\_additivity  
 category.h, 31  
 log4c\_category\_get\_appender  
 category.h, 31  
 log4c\_category\_get\_chainedpriority  
 category.h, 31  
 log4c\_category\_get\_name  
 category.h, 32  
 log4c\_category\_get\_priority  
 category.h, 32

---

log4c\_category\_info  
category.h, 32

log4c\_category\_is\_alert\_enabled  
category.h, 32

log4c\_category\_is\_crit\_enabled  
category.h, 32

log4c\_category\_is\_debug\_enabled  
category.h, 33

log4c\_category\_is\_error\_enabled  
category.h, 33

log4c\_category\_is\_fatal\_enabled  
category.h, 33

log4c\_category\_is\_info\_enabled  
category.h, 33

log4c\_category\_is\_notice\_enabled  
category.h, 34

log4c\_category\_is\_priority\_enabled  
category.h, 34

log4c\_category\_is\_trace\_enabled  
category.h, 34

log4c\_category\_is\_warn\_enabled  
category.h, 34

log4c\_category\_list  
category.h, 35

log4c\_category\_log  
category.h, 35

log4c\_category\_log\_locinfo  
category.h, 35

log4c\_category\_new  
category.h, 35

log4c\_category\_notice  
category.h, 36

log4c\_category\_print  
category.h, 36

log4c\_category\_set\_additivity  
category.h, 36

log4c\_category\_set\_appender  
category.h, 36

log4c\_category\_set\_priority  
category.h, 36

log4c\_category\_t  
category.h, 28

log4c\_category\_warn  
category.h, 37

log4c\_fini  
init.h, 37

log4c\_init  
init.h, 38

log4c\_layout\_delete  
layout.h, 40

log4c\_layout\_format  
layout.h, 40

log4c\_layout\_get  
layout.h, 40

log4c\_layout\_get\_name  
layout.h, 40

log4c\_layout\_get\_type  
layout.h, 40

log4c\_layout\_get\_udata  
layout.h, 41

log4c\_layout\_new  
layout.h, 41

log4c\_layout\_print  
layout.h, 41

log4c\_layout\_set\_type  
layout.h, 41

log4c\_layout\_set\_udata  
layout.h, 41

log4c\_layout\_t  
layout.h, 40

log4c\_layout\_type, 8

log4c\_layout\_type\_define  
layout.h, 39

log4c\_layout\_type\_get  
layout.h, 41

log4c\_layout\_type\_set  
layout.h, 42

log4c\_layout\_type\_t  
layout.h, 40

log4c\_layout\_types\_free  
layout.h, 42

log4c\_layout\_types\_print  
layout.h, 42

log4c\_load  
rc.h, 53

log4c\_location  
location\_info.h, 49

log4c\_location\_info\_t, 8

log4c\_logging\_event\_delete  
logging\_event.h, 50

log4c\_logging\_event\_new  
logging\_event.h, 50

log4c\_logging\_event\_t, 9

log4c\_major\_version  
version.h, 62

log4c\_micro\_version  
version.h, 62

log4c\_minor\_version  
version.h, 62

log4c\_priority\_level\_t  
priority.h, 52

log4c\_priority\_to\_int  
priority.h, 52

log4c\_priority\_to\_string  
priority.h, 52

log4c\_rc  
rc.h, 53

log4c\_rc\_t, 10

log4c\_rollingpolicy\_fini  
rollingpolicy.h, 56

log4c\_rollingpolicy\_get  
rollingpolicy.h, 56

log4c\_rollingpolicy\_get\_rfudata  
rollingpolicy.h, 56

log4c\_rollingpolicy\_get\_udata  
rollingpolicy.h, 56

- log4c\_rollingpolicy\_init
  - rollingpolicy.h, 57
- log4c\_rollingpolicy\_is\_triggering\_event
  - rollingpolicy.h, 57
- log4c\_rollingpolicy\_set\_type
  - rollingpolicy.h, 57
- log4c\_rollingpolicy\_set\_udata
  - rollingpolicy.h, 57
- log4c\_rollingpolicy\_t
  - rollingpolicy.h, 55
- log4c\_rollingpolicy\_type, 10
- log4c\_rollingpolicy\_type\_get
  - rollingpolicy.h, 58
- log4c\_rollingpolicy\_type\_set
  - rollingpolicy.h, 58
- log4c\_rollingpolicy\_type\_t
  - rollingpolicy.h, 55
- log4c\_stream2\_get\_flags
  - appender\_type\_stream2.h, 24
- log4c\_stream2\_get\_fp
  - appender\_type\_stream2.h, 24
- log4c\_stream2\_set\_flags
  - appender\_type\_stream2.h, 24
- log4c\_stream2\_set\_fp
  - appender\_type\_stream2.h, 25
- log4c\_version
  - version.h, 62
- logging\_event.h, 49
  - log4c\_logging\_event\_delete, 50
  - log4c\_logging\_event\_new, 50
- priority.h, 51
  - LOG4C\_PRIORITY\_ALERT, 52
  - LOG4C\_PRIORITY\_CRIT, 52
  - LOG4C\_PRIORITY\_DEBUG, 52
  - LOG4C\_PRIORITY\_ERROR, 52
  - LOG4C\_PRIORITY\_FATAL, 52
  - LOG4C\_PRIORITY\_INFO, 52
  - LOG4C\_PRIORITY\_NOTICE, 52
  - LOG4C\_PRIORITY\_NOTSET, 52
  - LOG4C\_PRIORITY\_TRACE, 52
  - LOG4C\_PRIORITY\_UNKNOWN, 52
  - LOG4C\_PRIORITY\_WARN, 52
  - log4c\_priority\_level\_t, 52
  - log4c\_priority\_to\_int, 52
  - log4c\_priority\_to\_string, 52
- ROLLINGPOLICY\_ROLLOVER\_ERR\_CAN\_LOG
  - rollingpolicy.h, 55
- rc.h, 53
  - log4c\_load, 53
  - log4c\_rc, 53
- rollingfile\_get\_current\_file\_size
  - appender\_type\_rollingfile.h, 20
- rollingfile\_make\_udata
  - appender\_type\_rollingfile.h, 20
- rollingfile\_udata\_get\_files\_prefix
  - appender\_type\_rollingfile.h, 20
- rollingfile\_udata\_get\_logdir
  - appender\_type\_rollingfile.h, 20
- rollingfile\_udata\_set\_files\_prefix
  - appender\_type\_rollingfile.h, 20
- rollingfile\_udata\_set\_logdir
  - appender\_type\_rollingfile.h, 21
- rollingfile\_udata\_set\_policy
  - appender\_type\_rollingfile.h, 21
- rollingpolicy.h, 54
  - log4c\_rollingpolicy\_fini, 56
  - log4c\_rollingpolicy\_get, 56
  - log4c\_rollingpolicy\_get\_rfudata, 56
  - log4c\_rollingpolicy\_get\_udata, 56
  - log4c\_rollingpolicy\_init, 57
  - log4c\_rollingpolicy\_is\_triggering\_event, 57
  - log4c\_rollingpolicy\_set\_type, 57
  - log4c\_rollingpolicy\_set\_udata, 57
  - log4c\_rollingpolicy\_t, 55
  - log4c\_rollingpolicy\_type\_get, 58
  - log4c\_rollingpolicy\_type\_set, 58
  - log4c\_rollingpolicy\_type\_t, 55
  - ROLLINGPOLICY\_ROLLOVER\_ERR\_CAN\_LOG, 55
- rollingpolicy\_sizewin\_udata\_t
  - rollingpolicy\_type\_sizewin.h, 59
- rollingpolicy\_type\_sizewin.h, 58
  - rollingpolicy\_sizewin\_udata\_t, 59
  - sizewin\_make\_udata, 59
  - sizewin\_udata\_set\_appender, 60
  - sizewin\_udata\_set\_file\_maxsize, 60
  - sizewin\_udata\_set\_max\_num\_files, 60
- sizewin\_make\_udata
  - rollingpolicy\_type\_sizewin.h, 59
- sizewin\_udata\_set\_appender
  - rollingpolicy\_type\_sizewin.h, 60
- sizewin\_udata\_set\_file\_maxsize
  - rollingpolicy\_type\_sizewin.h, 60
- sizewin\_udata\_set\_max\_num\_files
  - rollingpolicy\_type\_sizewin.h, 60
- version.h, 61
  - LOG4C\_MAJOR\_VERSION, 61
  - LOG4C\_MICRO\_VERSION, 61
  - LOG4C\_MINOR\_VERSION, 61
  - log4c\_major\_version, 62
  - log4c\_micro\_version, 62
  - log4c\_minor\_version, 62
  - log4c\_version, 62