

Package ‘log4r’

August 5, 2026

Type Package

Title A Fast and Lightweight Logging System for R, Based on 'log4j'

Version 0.5.0

Description Provides a fast, lightweight, object-oriented approach to logging based on the widely-emulated 'log4j' system and etymology. Loggers write messages to one or more destinations, such as the console, a file, or a remote server, using a format you control. Messages below the chosen severity threshold are discarded cheaply, so logging calls can be left in production code.

License Artistic-2.0

URL <https://github.com/r-lib/log4r>, <https://log4r.r-lib.org>

BugReports <https://github.com/r-lib/log4r/issues>

Imports cli, lifecycle, rlang

Suggests futile.logger, httr, jsonlite, knitr, lgr, logger, logging, microbenchmark, rlog, rmarkdown, rsyslog, testthat (>= 3.0.0), webfakes

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

LazyLoad yes

Collate 'appenders.R' 'logfuncs.R' 'deprecated.R' 'layouts.R' 'level.R' 'log4r-package.R' 'logger.R'

NeedsCompilation yes

Author John Myles White [aut, cph],
Kenton White [ctb],
Kirill Müller [ctb],
Aaron Jacobs [aut],
Posit Software, PBC [cph, fnd] (ROR: <<https://ror.org/03wc8by49>>),
Hadley Wickham [cre] (ORCID: <<https://orcid.org/0000-0003-4757-117X>>)

Maintainer Hadley Wickham <hadley@posit.co>
Repository CRAN
Date/Publication 2026-08-04 22:50:15 UTC

Contents

appenders	2
http_appender	3
layouts	4
level	5
logger	6
log_at	7
syslog_appender	8
tcp_appender	9
Index	10

appenders	<i>Send logs to their final destination with Appenders</i>
-----------	--

Description

In [log4j](#) etymology, **Appenders** are destinations where logs are written. Appenders have no control over formatting; this is controlled by the [Layout](#).

The most basic appenders write logs to the console or to a file; these are described below.

For implementing your own appenders, see [Details](#).

Usage

```
console_appender(layout = default_log_layout())  
  
file_appender(file, append = TRUE, layout = default_log_layout())
```

Arguments

- | | |
|--------|---|
| layout | A layout function taking a level parameter and additional arguments corresponding to the message. See layouts() . |
| file | The file to write messages to. |
| append | When TRUE, the file is not truncated when opening for the first time. |

Details

Appenders are implemented as functions with the interface `function(level, ...)`. These functions are expected to write their arguments to a destination and return `invisible(NULL)`.

Value

An appender: a function with the interface `function(level, ...)` that writes its arguments to a destination and returns `invisible(NULL)`.

See Also

[tcp_appender\(\)](#), [http_appender\(\)](#), [syslog_appender\(\)](#)

Examples

```
# The behaviour of an appender can be seen by using them directly; the
# following snippet will write the message to the console.
appender <- console_appender()
appender("INFO", "Input has length ", 0, ".")
```

http_appender	<i>Send logs over HTTP</i>
---------------	----------------------------

Description

Send logs in the body of HTTP requests. Responses with status code 400 or above will trigger errors.

Requires the `httr` package.

Usage

```
http_appender(url, method = "POST", layout = default_log_layout(), ...)
```

Arguments

<code>url</code>	The URL to submit messages to.
<code>method</code>	The HTTP method to use, usually "POST" or "GET".
<code>layout</code>	A layout function taking a <code>level</code> parameter and additional arguments corresponding to the message.
<code>...</code>	Further arguments passed on to httr::POST() .

Value

An appender: a function with the interface `function(level, ...)` that submits its arguments to `url` and returns `invisible(NULL)`.

See Also

[appenders](#) for more information on Appenders.

Examples

```
## Not run:
# POST messages to localhost.
appender <- http_appender("localhost")
appender("INFO", "Message.")

# POST JSON-encoded messages.
appender <- http_appender(
  "localhost", method = "POST", layout = default_log_layout(),
  httr::content_type_json()
)
appender("INFO", "Message.")

## End(Not run)
```

layouts

Format logs with Layouts

Description

In **log4j** etymology, **Layouts** are how **Appenders** control the format of messages. Most users will use one of the general-purpose layouts provided by the package:

- `default_log_layout()` formats messages much like the original log4j library. `simple_log_layout()` does the same, but omits the timestamp.
- `bare_log_layout()` emits only the log message, with no level or timestamp fields.
- `logfmt_log_layout()` and `json_log_layout()` format structured logs in the two most popular machine-readable formats.

For implementing your own layouts, see Details.

Usage

```
default_log_layout(time_format = "%Y-%m-%d %H:%M:%S")

simple_log_layout()

bare_log_layout()

logfmt_log_layout()

json_log_layout()
```

Arguments

`time_format` A valid format string for timestamps. See `base::strptime()`.

Details

Layouts return a function with the signature `function(level, ...)` that itself returns a single newline-terminated string. Anything that meets this interface can be passed as a layout to one of the existing [appenders](#).

`json_log_layout` requires the `jsonlite` package.

Value

A layout: a function with the interface `function(level, ...)` that returns a single newline-terminated string.

Examples

```
# The behaviour of a layout can be seen by using them directly:
simple <- simple_log_layout()
simple("INFO", "Input has length ", 0, ".")

with_timestamp <- default_log_layout()
with_timestamp("INFO", "Input has length ", 0, ".")

logfmt <- logfmt_log_layout()
logfmt("INFO", msg = "got input", length = 24)
```

level	<i>Set the logging threshold level for a logger dynamically</i>
-------	---

Description

It can sometimes be useful to change the logging threshold level at runtime. The `level()` accessor allows doing so.

Usage

```
level(x)

level(x) <- value

## S3 method for class 'logger'
level(x)

## S3 replacement method for class 'logger'
level(x) <- value

available.loglevels()
```

Arguments

`x` An object of class "logger".

`value` One of "DEBUG", "INFO", "WARN", "ERROR", or "FATAL".

Value

`level()` returns the logger's threshold and `level<-( )` returns the modified logger. `available.loglevels()` returns a named list of all thresholds.

Examples

```
lgr <- logger()
level(lgr) # Prints "INFO".
info(lgr, "This message is shown.")
level(lgr) <- "FATAL"
info(lgr, "This message is now suppressed.")
```

logger

Create a logger

Description

This is the main interface for configuring logging behaviour. We adopt the well-known **log4j** etymology: **Appenders** are destinations (e.g. the console or a file) where logs are written, and the **Layout** is the format of these logs.

Usage

```
logger(threshold = "INFO", appenders = console_appender())
```

Arguments

`threshold` The logging threshold, one of "DEBUG", "INFO", "WARN", "ERROR", or "FATAL". Logs with a lower severity than the threshold will be discarded.

`appenders` The logging appenders; both single appenders and a `list()` of them are supported. See **Appenders**.

Value

An object of class "logger".

See Also

Appenders and **Layouts** for information on controlling the behaviour of the logger object.

Examples

```
# By default, logs are written to the console at the "INFO" threshold.
logger <- logger()

log_info(logger, "Located nearest gas station.")
log_warn(logger, "Ez-Gas sensor network is not available.")
log_debug(logger, "Debug messages are suppressed by default.")
```

log_at	<i>Write logs at a given level</i>
--------	------------------------------------

Description

Write logs at a given level

Usage

```
log_at(logger, level, ...)

log_debug(logger, ...)

log_info(logger, ...)

log_warn(logger, ...)

log_error(logger, ...)

log_fatal(logger, ...)
```

Arguments

logger	An object of class "logger".
level	The desired severity, one of "DEBUG", "INFO", "WARN", "ERROR", or "FATAL". Messages with a lower severity than the logger threshold will be discarded.
...	One or more items to log.

Value

invisible(NULL), called for the side effect of writing to the logger's appenders.

Examples

```
logger <- logger()

log_at(logger, "WARN", "First warning from our code")
log_debug(logger, "Debugging our code")
log_info(logger, "Information about our code")
```

```
log_warn(logger, "Another warning from our code")
log_error(logger, "An error from our code")
log_fatal(logger, "I'm outta here")
```

syslog_appender	<i>Send logs to the local syslog</i>
-----------------	--------------------------------------

Description

Send messages to the local syslog. Requires the rsyslog package.

Usage

```
syslog_appender(identifier, layout = bare_log_layout(), ...)
```

Arguments

identifier	A string identifying the application.
layout	A layout function taking a level parameter and additional arguments corresponding to the message.
...	Further arguments passed on to rsyslog::open_syslog() .

Value

An appender: a function with the interface `function(level, ...)` that writes its arguments to the local syslog and returns `invisible(NULL)`.

See Also

[appenders](#) for more information on Appenders.

Examples

```
## Not run:
# Send messages to the local syslog.
appender <- syslog_appender("my-application")
appender("INFO", "Message.")

## End(Not run)
```

tcp_appender	<i>Send logs over TCP</i>
--------------	---------------------------

Description

Append messages to arbitrary TCP destinations.

Usage

```
tcp_appender(  
  host,  
  port,  
  layout = default_log_layout(),  
  timeout = getOption("timeout")  
)
```

Arguments

host	Hostname for the socket connection.
port	Port number for the socket connection.
layout	A layout function taking a level parameter and additional arguments corresponding to the message.
timeout	Timeout for the connection.

Value

An appender: a function with the interface `function(level, ...)` that writes its arguments to the socket and returns `invisible(NULL)`.

See Also

[appenders](#) for more information on Appenders, and [base::socketConnection\(\)](#) for the underlying connection object used by [tcp_appender\(\)](#).

Examples

```
## Not run:  
# Send messages to a listener on localhost.  
appender <- tcp_appender("localhost", port = 8080)  
appender("INFO", "Message.")  
  
## End(Not run)
```

Index

Appenders, [4](#), [6](#)
appenders, [2](#), [3](#), [5](#), [8](#), [9](#)
available.loglevels(level), [5](#)

bare_log_layout(layouts), [4](#)
bare_log_layout(), [4](#)
base::socketConnection(), [9](#)
base::strptime(), [4](#)

console_appender (appenders), [2](#)

default_log_layout(layouts), [4](#)
default_log_layout(), [4](#)

file_appender (appenders), [2](#)

http_appender, [3](#)
http_appender(), [3](#)
httr::POST(), [3](#)

json_log_layout(layouts), [4](#)
json_log_layout(), [4](#)

Layout, [2](#), [6](#)
Layouts, [6](#)
layouts, [4](#)
layouts(), [2](#)
level, [5](#)
level(), [5](#)
level<-(level), [5](#)
log_at, [7](#)
log_debug(log_at), [7](#)
log_error(log_at), [7](#)
log_fatal(log_at), [7](#)
log_info(log_at), [7](#)
log_warn(log_at), [7](#)
logfmt_log_layout(layouts), [4](#)
logfmt_log_layout(), [4](#)
logger, [6](#)

rsyslog::open_syslog(), [8](#)

simple_log_layout(layouts), [4](#)
simple_log_layout(), [4](#)
syslog_appender, [8](#)
syslog_appender(), [3](#)

tcp_appender, [9](#)
tcp_appender(), [3](#), [9](#)