Package: loggit2 (via r-universe)

August 25, 2024

Title Easy-to-Use, Dependencyless Logger

Description An easy-to-use 'ndjson' (newline-delimited 'JSON') logger. It provides a set of wrappers for base R's message(), warning(), and stop() functions that maintain identical functionality, but also log the handler message to an 'ndjson' log file. No change in existing code is necessary to use this package, and only a few additional adjustments are needed to fully utilize its potential.

Version 2.3.1

License MIT + file LICENSE

Depends R (>= 4.0)

Suggests knitr (>= 1.19), rmarkdown (>= 1.8), testthat (>= 3.0), utils

URL https://github.com/MEO265/loggit2, https://r-loggit.org/

BugReports https://github.com/MEO265/loggit2/issues

RoxygenNote 7.3.1

Encoding UTF-8

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Matthias Ollech [cre, aut], Ryan Price [fnd, aut]

Maintainer Matthias Ollech <ollech@gmx.com>

Repository CRAN

Date/Publication 2024-07-25 21:40:06 UTC

Contents

`convert_to_csv` *Write log to csv file*

Description

Creates a csv file from the ndjson log file.

Usage

```
convert_to_csv(
  file,
  logfile = get_logfile(),
  unsanitize = FALSE,
  last_first = FALSE,
  ...
)
```

Arguments

`file`	Path to write csv file to.
`logfile`	Path to log file to read from.
`unsanitize`	Should escaped special characters be unescaped?
`last_first`	Should the last log entry be the first row of the data frame?
`...`	Additional arguments to pass to `utils::write.csv()`.

Details

Unescaping of special characters can lead to unexpected results. Use `unsanitize = TRUE` with caution.

Value

Invisible NULL.

Examples

```
## Not run:
  convert_to_csv("my_log.csv")

  convert_to_csv("my_log.csv", logfile = "my_log.log", last_first = TRUE)

## End(Not run)
```

---

get_echo *Get echo*

Description

Get echo

Usage

```
get_echo()
```

Value

Logical. Are log messages echoed to stdout?

---

get_logfile *Get Log File*

Description

Return the log file that loggit() will write to by default.

Usage

```
get_logfile()
```

Value

The log file path.

Examples

```
get_logfile()
```

get_log_level *Get Log Level*

Description

Get Log Level

Usage

```
get_log_level()
```

Value

The log level.

get_timestamp_format *Get Timestamp Format*

Description

Get timestamp format for use in output logs.

Usage

```
get_timestamp_format()
```

Value

The timestamp format.

Examples

```
get_timestamp_format()
```

loggit *Log messages and R objects*

Description

Log messages and R objects to a ndjson log file.

Usage

```
loggit(
  log_lvl,
  log_msg,
  ...,
  echo = get_echo(),
  custom_log_lvl = FALSE,
  logfile = get_logfile(),
  ignore_log_level = FALSE
)
```

Arguments

log_lvl	Log level. A atomic vector of length one (usually character). Will be coerced to character.
log_msg	Log message. A atomic vector of length one (usually character). Will be coerced to character.
...	Named arguments, each a atomic vector of length one, you wish to log. Will be coerced to character. The names of the arguments are treated as column names in the log.
echo	Should the log entry (json) be echoed to stdout as well?
custom_log_lvl	Allow log levels other than "DEBUG", "INFO", "WARN", and "ERROR"?
logfile	Path of log file to write to.
ignore_log_level	Ignore the log level set by set_log_level()?

Value

Invisible NULL.

Examples

```
## Not run:
  loggit("DEBUG", "This is a message")

  loggit("INFO", "This is a message", echo = FALSE)

  loggit("CUSTOM", "This is a message of a custom log_lvl", custom_log_lvl = TRUE)
```

```
  loggit(
   "INFO", "This is a message", but_maybe = "you want more fields?",
    sure = "why not?", like = 2, or = 10, what = "ever"
  )

## End(Not run)
```

`message` *Message Log Handler*

Description

This function is identical to base R's `message`, but it includes logging of the exception message via `loggit()`.

Usage

```
message(..., domain = NULL, appendLF = TRUE, .loggit = NA, echo = get_echo())
```

Arguments

`...`	zero or more objects which can be coerced to character (and which are pasted together with no separator) or (for `message` only) a single condition object.
`domain`	see `gettext`. If `NA`, messages will not be translated, see also the note in `stop`.
`appendLF`	logical: should messages given as a character string have a newline appended?
`.loggit`	Should the condition message be added to the log? If `NA` the log level set by `set_log_level()` is used to determine if the condition should be logged.
`echo`	Should the log entry (json) be echoed to `stdout` as well?

Value

Invisible `NULL`.

See Also

Other handlers: `stop()`, `stopifnot()`, `warning()`

Examples

```
## Not run:
  message("Don't say such silly things!")

  message("Don't say such silly things!", appendLF = FALSE, echo = FALSE)

## End(Not run)
```

read_logs *Get log as* data.frame

Description

Returns a data.frame containing all the logs in the provided ndjson log file.

Usage

```
read_logs(logfile = get_logfile(), unsanitize = TRUE, last_first = FALSE)
```

Arguments

logfile	Path to log file to read from.
unsanitize	Should escaped special characters be unescaped?
last_first	Should the last log entry be the first row of the data frame?

Details

read_logs() returns a data.frame with the empty character columns "timestamp", "log_lvl" and "log_msg" if the log file has no entries.

Value

A data.frame, with the columns as the fields in the log file.

Examples

```
## Not run:
  read_logs()

  read_logs(last_first = TRUE)

## End(Not run)
```

rotate_logs *Rotate log file*

Description

Truncates the log file to the line count provided as rotate_lines.

Usage

```
rotate_logs(rotate_lines = 100000L, logfile = get_logfile())
```

Arguments

rotate_lines	The number of log entries to keep in the logfile.
logfile	Log file to truncate.

Value

Invisible NULL.

Examples

```
## Not run:
  rotate_logs()

  rotate_logs(rotate_lines = 0L)

  rotate_logs(rotate_lines = 1000L, logfile = "my_log.log")

## End(Not run)
```

---

set_echo *Set echo*

Description

Set echo

Usage

```
set_echo(echo = TRUE, confirm = TRUE)
```

Arguments

echo	Should log messages be echoed to stdout?
confirm	Print confirmation message of echo setting?

Value

Invisible the previous echo setting.

Examples

```
## Not run:
 set_echo(TRUE)
 set_echo(FALSE)

## End(Not run)
```

set_logfile *Set Log File*

Description

Set the log file that loggit will write to by default.

Usage

```
set_logfile(logfile = NULL, confirm = TRUE, create = TRUE)
```

Arguments

logfile	Absolut or relative path to log file. An attempt is made to convert the path into a canonical absolute form using normalizePath(). If NULL will set to <tmpdir>/loggit.log.
confirm	Print confirmation of log file setting?
create	Create the log file if it does not exist?

Details

No logs outside of a temporary directory will be written until this is set explicitly, as per CRAN policy. Therefore, the default behavior is to create a file named loggit.log in your system's temporary directory.

Value

Invisible the previous log file path.

Examples

```
## Not run:
  set_logfile("path/to/logfile.log")

## End(Not run)
```

set_log_level *Set Log Level*

Description

Set Log Level

Usage

```
set_log_level(level = "DEBUG", confirm = TRUE)
```

Arguments

level	Log level to set, as a string or integer.
confirm	Print confirmation message of log level?

Details

Log levels are as follows: DEBUG: 4 INFO: 3 WARNING: 2 ERROR: 1 NONE: 0

Value

Invisible the previous log level.

Examples

```
## Not run:
 set_log_level("DEBUG")
 set_log_level("INFO")

 set_log_level(4)
 set_log_level(3)

## End(Not run)
```

---

set_timestamp_format *Set Timestamp Format*

Description

Set timestamp format for use in output logs.

Usage

```
set_timestamp_format(ts_format = "%Y-%m-%dT%H:%M:%S%z", confirm = TRUE)
```

Arguments

ts_format	ISO date format.
confirm	Print confirmation message of timestamp format?

Details

This function performs no time format validations, but will echo out the current time in the provided format for manual validation.

This function provides no means of setting a timezone, and instead relies on the host system's time configuration to provide this. This is to enforce consistency across software running on the host.

Value

Invisible the previous timestamp format.

Examples

```
## Not run:
  set_timestamp_format("%Y-%m-%d %H:%M:%S")

## End(Not run)
```

`stop` *Stop Log Handler*

Description

This function is identical to base R's `stop`, but it includes logging of the exception message via `loggit()`.

Usage

```
stop(..., call. = TRUE, domain = NULL, .loggit = NA, echo = get_echo())
```

Arguments

`...`	zero or more objects which can be coerced to character (and which are pasted together with no separator) or a single condition object.
`call.`	logical, indicating if the call should become part of the error message.
`domain`	see `gettext`. If `NA`, messages will not be translated.
`.loggit`	Should the condition message be added to the log? If `NA` the log level set by `set_log_level()` is used to determine if the condition should be logged.
`echo`	Should the log entry (json) be echoed to `stdout` as well?

Value

No return value.

See Also

Other handlers: `message()`, `stopifnot()`, `warning()`

Examples

```
## Not run:
  stop("This is a completely false condition")

  stop("This is a completely false condition", echo = FALSE)

## End(Not run)
```

`stopifnot` *Conditional Stop Log Handler*

Description

This function is identical to base R's `stopifnot`, but it includes logging of the exception message via `loggit()`.

Usage

```
stopifnot(..., exprs, exprObject, local, .loggit = NA, echo = get_echo())
```

Arguments

`..., exprs`	any number of R expressions, which should each evaluate to (a logical vector of all) `TRUE`. Use *either* `...` *or* `exprs`, the latter typically an unevaluated expression of the form
	`{`
	`  expr1`
	`  expr2`
	`  ....`
	`}`
	Note that e.g., positive numbers are not `TRUE`, even when they are coerced to `TRUE`, e.g., inside `if(.)` or in arithmetic computations in R. If names are provided to `...`, they will be used in lieu of the default error message.

`exprObject`	alternative to `exprs` or `...`: an 'expression-like' object, typically an `expression`, but also a `call`, a `name`, or atomic constant such as `TRUE`.
`local`	(only when `exprs` is used:) indicates the `environment` in which the expressions should be evaluated; by default the one from where `stopifnot()` has been called.
`.loggit`	Should the condition message be added to the log? If `NA` the log level set by `set_log_level()` is used to determine if the condition should be logged.
`echo`	Should the log entry (json) be echoed to `stdout` as well?

See Also

Other handlers: `message()`, `stop()`, `warning()`

Examples

```
## Not run:
 stopifnot("This is a completely false condition" = FALSE)

 stopifnot(5L == 5L, "This is a completely false condition" = FALSE, echo = FALSE)

## End(Not run)
```

`warning` *Warning Log Handler*

Description

This function is identical to base R's `warning`, but it includes logging of the exception message via `loggit()`.

Usage

```
warning(
  ...,
  call. = TRUE,
  immediate. = FALSE,
  noBreaks. = FALSE,
  domain = NULL,
  .loggit = NA,
  echo = get_echo()
)
```

Arguments

`...`	zero or more objects which can be coerced to character (and which are pasted together with no separator) or a single condition object.
`call.`	logical, indicating if the call should become part of the warning message.
`immediate.`	logical, indicating if the call should be output immediately, even if `getOption("warn") <= 0`.
`noBreaks.`	logical, indicating as far as possible the message should be output as a single line when `options(warn = 1)`.
`domain`	see `gettext`. If `NA`, messages will not be translated, see also the note in `stop`.
`.loggit`	Should the condition message be added to the log? If `NA` the log level set by `set_log_level()` is used to determine if the condition should be logged.
`echo`	Should the log entry (json) be echoed to `stdout` as well?

Value

The warning message as `character` string, invisibly.

See Also

Other handlers: `message()`, `stop()`, `stopifnot()`

Examples

```
## Not run:
  warning("You may want to review that math")

  warning("You may want to review that math", immediate = FALSE, echo = FALSE)

## End(Not run)
```

`with_loggit` *Log any expressions*

Description

Log code without having to explicitly use the `loggit2` handlers. This is particularly useful for code that cannot be customized, e.g. from third-party packages.

Usage

```
with_loggit(
  exp,
  logfile = get_logfile(),
  echo = get_echo(),
  log_level = get_log_level()
)
```

Arguments

`exp`	An `expression` to evaluate.
`logfile`	Path of log file to write to.
`echo`	Should the log entry (json) be echoed to `stdout` as well?
`log_level`	The log level to use.

Details

If `loggit2` handlers are already used in the expression, this can lead to conditions being logged twice (in the same or different files).

Value

The result of the expression.

Examples

```
## Not run:
 x <- with_loggit(5L + 5L)

 with_loggit(base::message("Test log message"))

 with_loggit(base::warning("Test log message"), echo = FALSE, logfile = "my_log.log")

 x <- with_loggit({
    y <- 5L
    base::message("Test log message")
    base::warning("Test log message")
    1L + y
 })

## End(Not run)
```

Index