

Package: logrx (via r-universe)

October 20, 2026

Title A Logging Utility Focus on Clinical Trial Programming Workflows

Version 0.4.0.9000

Description A utility to facilitate the logging and review of R programs in clinical trial programming workflows.

License MIT + file LICENSE

URL <https://pharmaverse.github.io/logrx/>,
<https://github.com/pharmaverse/logrx>

BugReports <https://github.com/pharmaverse/logrx/issues>

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports dplyr (>= 1.0.0), magrittr, purrr, rlang, stats, tidyr,
sessioninfo (>= 1.2), tibble, digest, lifecycle, methods, yaml

Suggests testthat (>= 3.0.0), knitr, rmarkdown, withr, covr, pkgdown,
Tplyr, haven, lintr (>= 3.1.1), xml2, here, readr, rstudioapi,
tidyselect, renv

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 4.0.0)

Config/pak/sysreqs libicu-dev

Repository <https://pharmaverse.r-universe.dev>

Date/Publication 2026-10-20 12:57:38 UTC

RemoteUrl <https://github.com/pharmaverse/logrx>

RemoteRef HEAD

RemoteSha 76c8275860523d7c073796258a861f9f0d4421e9

Contents

approved	2
axecute	3
build_approved	4
log_config	6
log_init	7
log_remove	7
log_write	8
read_log_file	9
write_log_header	9
write_unapproved_functions	10
write_used_functions	10
Index	12

approved	<i>Approved packages and functions</i>
----------	--

Description

A dataset that stores approved packages and functions for use. Each row contains a library and function_name. This dataset is used to illustrate the data format to be stored in the log.rx.approved option.

Usage

approved

Format

A tibble with 6 rows and 2 variables:

- function_name** Name of the function
- library** Name of the package

Examples

logrx::approved

axecute

*Creation of a log and axecution of a file***Description**

axecute() executes a file and creates a log

Usage

```
axecute(
  file,
  log_name = NA,
  log_path = NA,
  include_rds = FALSE,
  quit_on_error = TRUE,
  to_report = c("messages", "output", "result"),
  show_repo_url = FALSE,
  error_on_unapproved = FALSE,
  extra_info = NA,
  ...
)
```

Arguments

file	String. Path to file to execute
log_name	String. Name of log file
log_path	String. Path to log file
include_rds	Boolean. Option to export log object as Rds file. Defaults to FALSE
quit_on_error	Boolean. Should the session quit with status 1 on error? Defaults to TRUE
to_report	String vector. Objects to optionally report, may include as many as necessary: • messages: any messages generated by program execution • output: any output generated by program execution • result: any result generated by program execution
show_repo_url	Boolean. Should the repository URLs be reported Defaults to FALSE
error_on_unapproved	Boolean. Should execution be stopped if unapproved packages are detected before running the script? Requires an approved list to be configured via <code>options(log.rx.approved)</code> . Defaults to FALSE. See the Logging Unapproved Package and Function Use vignette for details on setting up an approved package list.
extra_info	List. Objects to optionally add on to end of log in a special extra info section. List printed in YAML format. Optional
...	Not used

Value

NULL. Function is called for side effects. In non-interactive sessions, exits with status 1 if errors occur and `quit_on_error` is TRUE.

Handling Existing log.rx Environments

If a log.rx environment already exists when `axecute()` is called, the function will detect this and provide helpful guidance on how to resolve the issue. This typically occurs when a previous execution did not complete properly or the environment was not cleaned up.

In interactive mode, you will be presented with a menu offering two options:

1. Allow logrx to automatically remove the existing environment and proceed with execution
2. Cancel execution and manually remove the environment using `log_remove()` or by restarting your R session

In non-interactive mode (e.g., batch scripts), an error will be raised with detailed instructions on how to resolve the issue, including using `log_remove()` to programmatically clear the environment or restarting your R session.

See the [Execution vignette](#) for more details on handling existing environments.

Examples

```
dir <- tempdir()
text <- 'print("Hello, logrx-person!")'
fileConn <- file(file.path(dir, "hello.R"))
writeLines(text, fileConn)
close(fileConn)

axecute(file.path(dir, "hello.R"))

fileConn <- file(file.path(dir, "hello.Rmd"))
writeLines(text, fileConn)
close(fileConn)

axecute(file.path(dir, "hello.Rmd"))
```

build_approved

Build approved packages and functions

Description

A utility function to help you build your approved packages and functions list. This can be used by logrx to log unapproved use of packages and functions.

Usage

```
build_approved(
  pkg_list,
  file = NULL,
  yaml_prefix = TRUE,
  yaml_style = c("block", "inline")
)
```

Arguments

pkg_list	Named list of character vectors: • Name is the package name • Value is a character vector of approved functions, 'All', or 'all'
file	String. Path where the approved list will be written. If not specified, a tibble is returned. Default: NULL Permitted Files: .rds, .yaml, .yml
yaml_prefix	Logical. When writing a yaml file, should package keys be written with the package: prefix (e.g. package:base)? If FALSE, bare names are used (e.g. base). Ignored for .rds output. Default: TRUE
yaml_style	String. When writing a yaml file, "block" writes each function on its own line; "inline" writes functions as a flow sequence on one line. Ignored for .rds output. Default: "block"

Details

For more details see the vignette: `vignette("approved", package = "logrx")`

Value

If file is NULL, a tibble with two columns (library, function) and one row per function. If file is provided, the approved list is written to the specified file and nothing is returned.

Examples

```
approved_pkgs <- list(
  base = c("library", "mean"),
  dplyr = "All"
)

# build and return
build_approved(approved_pkgs)

# build and save
dir <- tempdir()
build_approved(approved_pkgs, file.path(dir, "approved.rds"))
```

log_config	<i>Configuration of the log.rx environment</i>
------------	--

Description

log_config() initialises the log.rx environment, adds its attributes, and sets them

Usage

```
log_config(file = NA, log_name = NA, log_path = NA, extra_info = NA)
```

Arguments

file	String. Path to file executed. Optional
log_name	String. Name of log file. Optional
log_path	String. Path to log file. Optional
extra_info	List. Objects to add on to end of log in a special extra info section. Optional

Value

Nothing

Examples

```
dir <- tempdir()
text <- 'print("Hello, Timberperson!")'
fileConn <- file(file.path(dir, "hello.R"))
writeLines(text, fileConn)
close(fileConn)

file <- file.path(dir, "hello.R")

# Initialise and configure the log.rx environment
log_config(file)

# Run the script and record results, outputs, messages, errors, and warnings
logrx::run_safely_loudly(file)

# Write the log
log_write(file)
```

log_init	<i>Initialization of the log.rx environment</i>
----------	---

Description

log_init() initialises the log.rx environment

Usage

```
log_init()
```

Value

Nothing

Examples

```
# Initialise the log.rx environment
log_init()

# Remove the log.rx environment
log_remove()
```

log_remove	<i>log.rx object removal</i>
------------	------------------------------

Description

log_remove() removes the log.rx object by setting options("log.rx") to NULL

Usage

```
log_remove()
```

Value

Nothing

Examples

```
# Initialise the log.rx environment
log_init()

# Remove the log.rx environment
log_remove()
```

log_write

*Formatting and writing of the log.rx object to a log file***Description**

log_write() gets and formats the content of the log.rx before writing it to a log file

Usage

```
log_write(
  file = NA,
  remove_log_object = TRUE,
  show_repo_url = FALSE,
  include_rds = FALSE,
  extra_info = NA,
  to_report = c("messages", "output", "result")
)
```

Arguments

file	String. Path to file executed
remove_log_object	Boolean. Should the log object be removed after writing the log file? Defaults to TRUE
show_repo_url	Boolean. Should the repo URLs be reported Defaults to FALSE
include_rds	Boolean. Option to export log object as Rds file. Defaults to FALSE
extra_info	List. Objects to add on to end of log in a special extra info section. Optional
to_report	String vector. Objects to optionally report; additional information in axecute

Value

Nothing

Examples

```
dir <- tempdir()
text <- 'print("Hello, Timberperson!")'
fileConn <- file(file.path(dir, "hello.R"))
writeLines(text, fileConn)
close(fileConn)

file <- file.path(dir, "hello.R")

# Initialise and configure the log.rx environment
log_config(file)

# Run the script and record results, outputs, messages, errors, and warnings
```

```
logrx:::run_safely_loudly(file)

# Write the log
log_write(file)
```

read_log_file	<i>Read and parse logrx file</i>
---------------	----------------------------------

Description

Read and parse logrx file

Usage

```
read_log_file(file)
```

Arguments

file	String. Path to a logrx log file
------	----------------------------------

Value

Tibble. Object that includes nested and parsed content

Examples

```
## Not run:
read_log_file(previous_log_filepath)

## End(Not run)
```

write_log_header	<i>Formatting of log file section headers</i>
------------------	---

Description

write_log_header formats a string and returns it as a formatted log file section header

Usage

```
write_log_header(title_string)
```

Arguments

title_string	String. Used as section title
--------------	-------------------------------

Value

Vector of strings. Formatted log file section header

Examples

```
## Not run:  
write_log_header("Section Header")  
  
## End(Not run)
```

write_unapproved_functions

Formats and returns a vector of unapproved functions

Description

write_unapproved_functions() gets log.rx unapproved_packages_functions attribute, formats and returns the list of unapproved functions

Usage

```
write_unapproved_functions()
```

Value

Formatted vector of unapproved functions

Examples

```
## Not run:  
write_unapproved_functions()  
  
## End(Not run)
```

write_used_functions *Formats and returns a vector of used package functions*

Description

write_used_functions() gets log.rx used_packages_functions attribute, formats and returns the list of used package functions

Usage

```
write_used_functions()
```

Value

Formatted vector of used package functions

Examples

```
## Not run:  
write_used_functions()  
  
## End(Not run)
```

Index

* **datasets**

approved, [2](#)

approved, [2](#)

axecute, [3](#), [8](#)

build_approved, [4](#)

log_config, [6](#)

log_init, [7](#)

log_remove, [7](#)

log_write, [8](#)

read_log_file, [9](#)

write_log_header, [9](#)

write_unapproved_functions, [10](#)

write_used_functions, [10](#)