

# Package ‘logrx’

October 13, 2022

**Title** A Logging Utility Focus on Clinical Trial Programming Workflows

**Version** 0.1.1

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

**License** MIT + file LICENSE

**URL** <https://github.com/atorus-research/logrx>

**BugReports** <https://github.com/atorus-research/logrx/issues>

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.1

**Imports** dplyr, magrittr, purrr, rlang, stats, tidyr, stringr, miniUI,  
rstudioapi, shiny, sessioninfo (>= 1.2), stringi, waiter,  
tibble

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown, withr, covr, pkgdown,  
Tplyr, haven, here

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Depends** R (>= 3.5.0)

**NeedsCompilation** no

**Author** Nathan Kosiba [aut, cre] (<<https://orcid.org/0000-0001-5359-4234>>),  
Thomas Bermudez [aut],  
Ben Straub [aut],  
Michael Rimler [aut],  
Nicholas Masel [aut],  
GSK/Atorus JPT [cph, fnd]

**Maintainer** Nathan Kosiba <Nathan.Kosiba@atorusresearch.com>

**Repository** CRAN

**Date/Publication** 2022-09-12 13:20:02 UTC

**R topics documented:**

approved . . . . .	2
approved_functions_init . . . . .	3
axecute . . . . .	4
build_approved . . . . .	4
get_file_path . . . . .	5
get_library . . . . .	6
get_logrx_metadata . . . . .	6
get_log_element . . . . .	7
get_masked_functions . . . . .	7
get_session_info . . . . .	8
get_unapproved_use . . . . .	8
get_used_functions . . . . .	9
logrxAddin . . . . .	9
log_cleanup . . . . .	10
log_config . . . . .	10
log_init . . . . .	11
log_remove . . . . .	11
log_write . . . . .	11
loudly . . . . .	12
run_safely_loudly . . . . .	12
set_log_element . . . . .	13
set_log_name_path . . . . .	13
write_errors . . . . .	14
write_file_name_path . . . . .	14
write_log_element . . . . .	14
write_log_header . . . . .	15
write_masked_functions . . . . .	15
write_messages . . . . .	16
write_metadata . . . . .	16
write_output . . . . .	16
write_result . . . . .	17
write_session_info . . . . .	17
write_unapproved_functions . . . . .	17
write_used_functions . . . . .	18
write_warnings . . . . .	18
<b>Index</b>	<b>19</b>

---

 approved

*Approved packages and functions*


---

**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 logrx.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

---

approved\_functions\_init  
*Initialises the logrx.approved option*

---

**Description**

Defaults to working directory. This should point to the location of the dataframe storing approved packages and functions.

**Usage**

approved\_functions\_init()

**Details**

See ?approved for an example dataframe.

**Value**

Nothing

---

axecute	<i>Create a log and run a file</i>
---------	------------------------------------

---

### Description

Create a log and run a file

### Usage

```
axecute(
  file,
  log_name = NA,
  log_path = NA,
  remove_log_object = TRUE,
  quit_on_error = TRUE,
  to_report = c("messages", "output", "result")
)
```

### Arguments

file	File path of file to run
log_name	Name of log file
log_path	Path to output log to
remove_log_object	Should the log object be removed after writing, defaults to TRUE
quit_on_error	Should the session quit with status of 1 with error, defaults to TRUE
to_report	toggle for optional reporting objects. choices outlined below, 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

### Value

0 if there are no errors and 1 if any error

---

build_approved	<i>Build approved packages and functions tibble</i>
----------------	---

---

### Description

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

**Usage**

```
build_approved(pkg_list, file = NULL)
```

**Arguments**

pkg_list	A named list of character vectors where the name is the package name with a character vector of approved functions or 'All'
file	Name of file where the approved tibble will be read to. If not specified, the tibble is returned. Default: NULL Permitted Files: .RDS

**Details**

For more details see the help vignette:

```
vignette("approved", package = "logrx")
```

**Value**

A tibble with two columns (library, function) and one row per function

**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"))
```

---

get\_file\_path

*Gets full path of file being run*

---

**Description**

Gets full path of file being run

**Usage**

```
get_file_path(file = NA, normalize = TRUE)
```

**Arguments**

file                File path of file being run, optional  
 normalize        If the returned path should be normalized

**Value**

full path of file being run

**Examples**

```
get_file_path()
```

---

get_library	<i>Add libraries to functions</i>
-------------	-----------------------------------

---

**Description**

Each script should be independent so we can use the search path since this would be just for this script. This must also be run after script execution.

**Usage**

```
get_library(df)
```

**Arguments**

df                dataframe containing variables function\_name and SYMBOL\_PACKAGE

**Value**

tibble that includes library

---

get_logrx_metadata	<i>Returns named list of logrx metadata attributes</i>
--------------------	--

---

**Description**

Returns named list of logrx metadata attributes

**Usage**

```
get_logrx_metadata()
```

**Value**

Named list of logrx package metadata attributes

**Examples**

```
get_logrx_metadata()
```

---

<code>get_log_element</code>	<i>Gets the value of a named element in the log.rx environment</i>
------------------------------	--

---

**Description**

Gets the value of a named element in the log.rx environment

**Usage**

```
get_log_element(el_key)
```

**Arguments**

`el_key`            the key of the element in log.rx to be fetched

**Value**

Value of corresponding element from log.rx environment

---

<code>get_masked_functions</code>	<i>Returns named list of masked functions</i>
-----------------------------------	---

---

**Description**

Returns named list of masked functions

**Usage**

```
get_masked_functions()
```

**Value**

Named list of masked functions, source package, and what they mask

**Examples**

```
get_masked_functions()
```

---

get_session_info	<i>Returns Session Info</i>
------------------	-----------------------------

---

**Description**

Returns Session Info

**Usage**

```
get_session_info()
```

**Value**

Formatted Session Info

**Examples**

```
get_session_info()
```

---

get_unapproved_use	<i>Get unapproved packages and functions used</i>
--------------------	---

---

**Description**

Compare two dataframes that contain approved and used packages and functions.

**Usage**

```
get_unapproved_use(approved_packages, used_packages)
```

**Arguments**

approved\_packages      dataframe containing variables function\_name and library  
used\_packages      dataframe containing variables function\_name and library

**Value**

tibble that includes packages and functions used, but not approved



---

get_used_functions	<i>Get functions used within a file</i>
--------------------	---

---

**Description**

Get functions used within a file

**Usage**

```
get_used_functions(file)
```

**Arguments**

file	File path of file to run
------	--------------------------

**Value**

tibble with library and function\_name

**Examples**

```
## Not run:  
file <- "ex1.R"  
get_functions_used(file)  
  
## End(Not run)
```

---

logrxAddin	<i>logrxAddin Code needed to create the miniUI for the Addin</i>
------------	--

---

**Description**

logrxAddin Code needed to create the miniUI for the Addin

**Usage**

```
logrxAddin()
```

**Value**

returns miniUI Addin to batch submit r files and create logs around them

---

log_cleanup	<i>Cleans up log and does checks against elements</i>
-------------	---

---

**Description**

Cleans up log and does checks against elements

**Usage**

```
log_cleanup()
```

**Value**

List of non-NA elements and their value in log.rx environment

---

log_config	<i>Configures the log.rx environment</i>
------------	--

---

**Description**

Configures the log.rx environment

**Usage**

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

**Arguments**

file	File path of file being run, optional
log_name	The log name
log_path	The log path

**Value**

Nothing

---

log_init	<i>Initialises the log.rx environment</i>
----------	---

---

**Description**

Initialises the log.rx environment

**Usage**

```
log_init()
```

**Value**

Nothing

---

log_remove	<i>Remove the log.rx environment by setting options("log.rx") to NULL</i>
------------	---

---

**Description**

Remove the log.rx environment by setting options("log.rx") to NULL

**Usage**

```
log_remove()
```

**Value**

Nothing

---

log_write	<i>Write the formatted log.rx to a file</i>
-----------	---

---

**Description**

Write the formatted log.rx to a file

**Usage**

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

**Arguments**

file            File path of file being run  
 remove\_log\_object    Should the log object be removed after writing, defaults to TRUE  
 to\_report       toggle for optional reporting objects, additional information in [axecute](#)

**Value**

Nothing

---

loudly	<i>Capture errors, warnings, messages, and a stream output from executing R code</i>
--------	--

---

**Description**

Capture errors, warnings, messages, and a stream output from executing R code

**Usage**

loudly(code)

**Arguments**

code            All code to be run loudly

**Value**

Wrapped function returns a list with components result, stream, messages and warnings.

---

run_safely_loudly	<i>Safely run an R script and record results, outputs, messages, errors, warnings</i>
-------------------	---

---

**Description**

Safely run an R script and record results, outputs, messages, errors, warnings

**Usage**

run\_safely\_loudly(file)

**Arguments**

file            File to run

**Value**

Nothing

---

set_log_element	<i>Adds values to existing named elements in the log.rx environment</i>
-----------------	---

---

**Description**

Adds values to existing named elements in the log.rx environment

**Usage**

```
set_log_element(el_key, el_value)
```

**Arguments**

el_key	the key of the element in log.rx to be updated
el_value	the value to be added to the log.rx element

**Value**

Nothing

---

set_log_name_path	<i>Set the log name and path</i>
-------------------	----------------------------------

---

**Description**

1. As the name and path if supplied
2. As the file name with .log extension and path if specified or if they can be determined by the function
3. As logrx\_log.log and . if none of the above are specified

**Usage**

```
set_log_name_path(log_name = NA, log_path = NA)
```

**Arguments**

log_name	The log name
log_path	The log path

**Value**

Nothing

---

write_errors	<i>Format errors attribute for writing</i>
--------------	--

---

**Description**

Format errors attribute for writing

**Usage**

```
write_errors()
```

**Value**

A formatted vector of errors

---

write_file_name_path	<i>Format file name and path for writing</i>
----------------------	--

---

**Description**

Format file name and path for writing

**Usage**

```
write_file_name_path()
```

**Value**

A vector of file name and path prefixed

---

write_log_element	<i>Generic function to format log.rx elements for writing</i>
-------------------	---

---

**Description**

Generic function to format log.rx elements for writing

**Usage**

```
write_log_element(el_key, prefix = NULL)
```

**Arguments**

el_key	the key of the element in log.rx to be fetched
prefix	string to be placed before element value during formatting

**Value**

formatted element including prefix

---

write\_log\_header      *Generic function to format log section headers*

---

**Description**

Generic function to format log section headers

**Usage**

```
write_log_header(title_string)
```

**Arguments**

title\_string      String to be used as section title

**Value**

A vector with the header including title

---

write\_masked\_functions  
*Format masked functions attribute for writing*

---

**Description**

Format masked functions attribute for writing

**Usage**

```
write_masked_functions()
```

**Value**

A formatted vector of masked functions

---

write_messages	<i>Format messages attribute for writing</i>
----------------	--

---

**Description**

Format messages attribute for writing

**Usage**

```
write_messages()
```

**Value**

A formatted vector of messages

---

write_metadata	<i>Format log.rx's metadata attributes for writing</i>
----------------	--

---

**Description**

Format log.rx's metadata attributes for writing

**Usage**

```
write_metadata()
```

**Value**

A vector of log.rx's metadata attributes

---

write_output	<i>Format output attribute for writing</i>
--------------	--

---

**Description**

Format output attribute for writing

**Usage**

```
write_output()
```

**Value**

A formatted vector of output



---

write_result	<i>Format result attribute for writing</i>
--------------	--

---

**Description**

Format result attribute for writing

**Usage**

```
write_result()
```

**Value**

A formatted vector of results

---

write_session_info	<i>Format log.rx's session info attribute for writing</i>
--------------------	---

---

**Description**

Format log.rx's session info attribute for writing

**Usage**

```
write_session_info()
```

**Value**

A vector of log.rx's session info attribute

---

write_unapproved_functions	<i>Format unapproved functions attribute for writing</i>
----------------------------	--

---

**Description**

Format unapproved functions attribute for writing

**Usage**

```
write_unapproved_functions()
```

**Value**

A formatted vector of unapproved functions

**Examples**

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

---

write\_used\_functions    *Format used functions attribute for writing*

---

**Description**

Format used functions attribute for writing

**Usage**

```
write_used_functions()
```

**Value**

A formatted vector of used functions

**Examples**

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

---

write\_warnings        *Format warnings attribute for writing*

---

**Description**

Format warnings attribute for writing

**Usage**

```
write_warnings()
```

**Value**

A formatted vector of warnings

# Index

- \* **datasets**
  - approved, [2](#)
- approved, [2](#)
- approved\_functions\_init, [3](#)
- axecute, [4](#), [12](#)
  
- build\_approved, [4](#)
  
- get\_file\_path, [5](#)
- get\_library, [6](#)
- get\_log\_element, [7](#)
- get\_logrx\_metadata, [6](#)
- get\_masked\_functions, [7](#)
- get\_session\_info, [8](#)
- get\_unapproved\_use, [8](#)
- get\_used\_functions, [9](#)
  
- log\_cleanup, [10](#)
- log\_config, [10](#)
- log\_init, [11](#)
- log\_remove, [11](#)
- log\_write, [11](#)
- logrxAddin, [9](#)
- loudly, [12](#)
  
- run\_safely\_loudly, [12](#)
  
- set\_log\_element, [13](#)
- set\_log\_name\_path, [13](#)
  
- write\_errors, [14](#)
- write\_file\_name\_path, [14](#)
- write\_log\_element, [14](#)
- write\_log\_header, [15](#)
- write\_masked\_functions, [15](#)
- write\_messages, [16](#)
- write\_metadata, [16](#)
- write\_output, [16](#)
- write\_result, [17](#)
- write\_session\_info, [17](#)
  
- write\_unapproved\_functions, [17](#)
- write\_used\_functions, [18](#)
- write\_warnings, [18](#)