

Package: roxyreqs (via r-universe)

July 16, 2026

Title 'roxygen2'-Style Metadata for Test Cases and Function Documentation

Version 1.3.0

Description Extends 'roxygen2' to support '@meta' tags for documenting 'testthat' test cases and function specifications. Includes a custom 'JUnit' reporter that exports test metadata as XML properties and validation functions to ensure all exported functions and tests contain required tags. Designed for traceability between requirements and tests in regulated industries such as pharma and finance.

License GPL (>= 3)

URL <https://github.com/mnlang/roxyreqs>

BugReports <https://github.com/mnlang/roxyreqs/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE, roclets = c("`rd", `` namespace"))

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports brio, cli, R6, rlang, roxygen2 (>= 7.0.0), stringr, testthat (>= 3.2.0), withr

Suggests spelling, xml2

Config/testthat/edition 3

Language en-US

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev

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

Date/Publication 2026-07-06 12:24:27 UTC

RemoteUrl <https://github.com/mnlang/roxyreqs>

RemoteRef HEAD

RemoteSha 43f6b482e443570e966d13eb6ba1c76d0072911d

Contents

check_meta	2
JUnitReporterMeta	4
meta_tags	7
move_macro_first	8
parse_rd_meta	9
Index	10

check_meta	<i>Check Metadata Documentation</i>
------------	-------------------------------------

Description

A suite of functions to validate metadata documentation in Rd and test files.

Meta tags follow the format @meta key value where key can be any user-defined identifier. Built-in tags for function documentation include id, compliance_risk, and implementation_complexity. For test files, the default required tags are author, reviewer, review_date, and description. Custom tags can be added and validated by passing them to the check_rd_tag_exist or check_test_tag_exist parameters.

This function combines multiple checks to ensure metadata in your Rd and test files meets required standards for completeness, uniqueness, and documentation quality. The following checks are included:

- `check_meta_rd()`: Checks for missing metadata tags in Rd files.
- `check_meta_unique_ids()`: Ensures the 'Id' metadata tag in Rd files is unique.
- `check_meta_undocumented_test()`: Validates that all test_that test cases are properly documented.
- `check_meta_test()`: Verifies that test case documentation contains all required metadata tags.

Usage

```
check_meta(
  files_rd = list.files("man", pattern = "*.Rd", full.names = TRUE),
  files_test = list.files("tests/testthat", pattern = "^test-.*\\.R$", full.names =
    TRUE),
  check_rd_tag_exist = c("Id", "Compliance Risk", "Implementation Complexity",
    "Test Scope"),
  check_test_tag_exist = c("author", "reviewer", "review_date", "description"),
  check_rd_tag_unique = "Id",
  exclude_package_rd = TRUE,
  concise = TRUE,
  print_error = FALSE
)
```

```

check_meta_test(
  files_test = list.files("tests/testthat", pattern = "^test-.*\\.R$", full.names =
    TRUE),
  check_test_tag_exist = c("author", "reviewer", "review_date", "description"),
  concise = FALSE,
  print_error = FALSE
)

check_meta_undocumented_test(
  files_test = list.files("tests/testthat", pattern = "^test-.*\\.R$", full.names =
    TRUE),
  concise = FALSE,
  print_error = FALSE
)

check_meta_rd(
  files_rd = list.files("man", pattern = "*.Rd", full.names = TRUE),
  check_rd_tag_exist = c("Id", "Compliance Risk", "Implementation Complexity",
    "Test Scope"),
  exclude_package_rd = TRUE,
  concise = FALSE,
  print_error = FALSE
)

check_meta_unique_ids(
  files_rd = list.files("man", pattern = "*.Rd", full.names = TRUE),
  check_rd_tag_unique = "Id",
  exclude_package_rd = TRUE,
  concise = FALSE,
  print_error = FALSE
)

```

Arguments

<code>files_rd</code>	character; A vector of paths to .Rd files to be checked.
<code>files_test</code>	character; A vector of paths to test files to be checked.
<code>check_rd_tag_exist</code>	character; A vector of metadata tag names in Rd files to check for completeness.
<code>check_test_tag_exist</code>	character; A vector of metadata tag names in test files to check for completeness.
<code>check_rd_tag_unique</code>	character; A single metadata tag in Rd files to check for uniqueness. Default is "Id".
<code>exclude_package_rd</code>	logical; Whether to exclude package-level Rd files (i.e., files ending with "-package.Rd"). Default is TRUE.
<code>concise</code>	logical; Whether to print only failures instead of detailed output. Default is TRUE.

`print_error` logical; Whether to print error messages. This is useful for CI/CD pipelines to produce errors when issues are found. Default is FALSE.

Value

- `check_meta()`: A named list with results from each sub-check (invisibly).

Meta information

Id: URS01.FS005

Compliance Risk: low

Implementation Complexity: low

Test Scope: low

JUnitReporterMeta	<i>Test reporter: summary of errors in JUnit XML format.</i>
-------------------	--

Description

This reporter includes detailed results about each test and summaries, written to a file (or stdout) in JUnit XML format. This can be read by the Jenkins Continuous Integration System to report on a dashboard etc. Requires the *xml2* package.

Details

To fit into the JUnit structure, `context()` becomes the `<testsuite>` name as well as the base of the `<testcase>` classname. The `test_that()` name becomes the rest of the `<testcase>` classname. The deparsed `expect_that()` call becomes the `<testcase>` name. On failure, the message goes into the `<failure>` node message argument (first line only) and into its text content (full message).

Execution time and some other details are also recorded.

Tests without `@meta roxygen` blocks produce test cases with an empty `<properties/>` node. Use `check_meta_test()` to validate that all required tags are present.

References for the JUnit XML format: <http://llg.cubic.org/docs/junit/>

Super classes

`testthat::Reporter` -> `testthat::JUnitReporter` -> `JUnitReporterMeta`

Public fields

`results` Test results.

`timer` Process time tracker.

`doc` XML document object.

`errors` Error count for current suite.

`failures` Failure count for current suite.

skipped Skip count for current suite.
tests Test count for current suite.
root XML root node (<testsuites>).
suite Current XML suite node (<testsuite>).
suite_time Elapsed time for current suite.
file_name Current test file path.
roxy_blocks Parsed roxygen2 blocks for current file.

Methods

Public methods:

- [JUnitReporterMeta\\$elapsed_time\(\)](#)
- [JUnitReporterMeta\\$reset_suite\(\)](#)
- [JUnitReporterMeta\\$start_reporter\(\)](#)
- [JUnitReporterMeta\\$start_file\(\)](#)
- [JUnitReporterMeta\\$start_test\(\)](#)
- [JUnitReporterMeta\\$start_context\(\)](#)
- [JUnitReporterMeta\\$end_context\(\)](#)
- [JUnitReporterMeta\\$add_result\(\)](#)
- [JUnitReporterMeta\\$end_reporter\(\)](#)
- [JUnitReporterMeta\\$clone\(\)](#)

Method `elapsed_time()`: Return elapsed time since last call and reset timer.

Usage:

`JUnitReporterMeta$elapsed_time()`

Returns: Elapsed time in seconds.

Method `reset_suite()`: Reset suite counters to zero.

Usage:

`JUnitReporterMeta$reset_suite()`

Method `start_reporter()`: Initialize the reporter, create XML document.

Usage:

`JUnitReporterMeta$start_reporter()`

Method `start_file()`: Start processing a test file. Parses roxygen2 blocks.

Usage:

`JUnitReporterMeta$start_file(file)`

Arguments:

`file` Path to the test file.

Method `start_test()`: Called at the start of each test.

Usage:

JUnitReporterMeta\$start_test(context, test)

Arguments:

context Test context name.

test Test name.

Method start_context(): Start a new test suite context.

Usage:

JUnitReporterMeta\$start_context(context)

Arguments:

context Context name.

Method end_context(): Finalize a test suite context. Writes counts and time.

Usage:

JUnitReporterMeta\$end_context(context)

Arguments:

context Context name.

Method add_result(): Add a test result. Matches @meta tags from roxygen2 blocks and writes them as XML properties.

Usage:

JUnitReporterMeta\$add_result(context, test, result)

Arguments:

context Context name.

test Test name.

result A testthat expectation object.

Method end_reporter(): Write the XML document to the output destination.

Usage:

JUnitReporterMeta\$end_reporter()

Method clone(): The objects of this class are cloneable with this method.

Usage:

JUnitReporterMeta\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Meta information

Id: URS01.FS007

Compliance Risk: low

Implementation Complexity: medium

Test Scope: medium

Examples

```
## Not run:
testthat::test_local(path = "inst/example.pkg", reporter = JunitReporterMeta)

## End(Not run)
```

meta_tags

*Parse and format roxygen2 meta tags***Description**

S3 methods that extend roxygen2 to support @meta tags in function and test documentation. Tags follow the format @meta key value where key can be any user-defined identifier (underscores become spaces, title-cased in output).

Usage

```
## S3 method for class 'roxy_tag_meta'
roxy_tag_parse(x)

## S3 method for class 'roxy_tag_meta'
roxy_tag_rd(x, base_path, env)

## S3 method for class 'rd_section_meta'
merge(x, y, ...)

## S3 method for class 'rd_section_meta'
format(x, ...)
```

Arguments

x	A roxygen2 tag, rd_section, or Rd object depending on the method.
base_path	Base path of the package (used by roxygen2 dispatch).
env	Package environment (used by roxygen2 dispatch).
y	A second rd_section_meta object to merge with x.
...	Additional arguments passed to methods.

Details

format.rd_section_meta automatically computes a Test Scope value from Compliance Risk and Implementation Complexity using an internal scopemap matrix. Tags are reordered with Id, Compliance Risk, Implementation Complexity, and Test Scope first, followed by any custom tags.

Meta information

Id: URS01.FS011
Compliance Risk: low
Implementation Complexity: low
Test Scope: low

move_macro_first	<i>Move macro (meta) fragment right after arguments</i>
------------------	---

Description

Move macro (meta) fragment right after arguments

Usage

```
move_macro_first(x, rd_tag = "\\title", offset = 1L)
```

Arguments

x	An Rd object as returned by <code>tools::parse_Rd()</code>
rd_tag	character; Rd tag name after which macro section shall be moved
offset	integer; Offset index, so that macro block is moved after rd_tag plus following text.

Value

transformed Rd object

Meta information

Id: URS01.FS012
Compliance Risk: low
Implementation Complexity: low
Test Scope: low

parse_rd_meta	<i>Parse Metadata from Rd Files</i>
---------------	-------------------------------------

Description

Parse Metadata from Rd Files

Usage

```
parse_rd_meta(filepath)
```

Arguments

filepath Character array specifying paths to Rd-files.

Value

data.frame containing one column for each meta tag and one row per filepath.

Meta information

Id: URS01.FS006

Compliance Risk: low

Implementation Complexity: low

Test Scope: low

Index

* **reporters**

JUnitReporterMeta, 4

check_meta, 2

check_meta_rd (check_meta), 2

check_meta_rd(), 2

check_meta_test (check_meta), 2

check_meta_test(), 2, 4

check_meta_undocumented_test

(check_meta), 2

check_meta_undocumented_test(), 2

check_meta_unique_ids (check_meta), 2

check_meta_unique_ids(), 2

format.rd_section_meta (meta_tags), 7

JUnitReporterMeta, 4

merge.rd_section_meta (meta_tags), 7

meta_tags, 7

move_macro_first, 8

parse_rd_meta, 9

roxy_tag_parse.roxy_tag_meta

(meta_tags), 7

roxy_tag_rd.roxy_tag_meta (meta_tags), 7

testthat::JUnitReporter, 4

testthat::Reporter, 4