

# Package ‘cnd’

October 8, 2026

**Title** Create and Register Conditions

**Version** 0.2.0

**Description** An interface for creating new condition generators objects.  
Generators are special functions that can be saved in registries and linked to other functions. Utilities for documenting your generators, and new conditions is provided for package development.

**License** MIT + file LICENSE

**BugReports** <https://github.com/jmbarbone/cnd/issues>

**URL** <https://jmbarbone.github.io/cnd/>, <https://github.com/jmbarbone/cnd>

**Encoding** UTF-8

**Language** en-US

**Depends** R (>= 4.1)

**Suggests** cli (>= 3.6.5), here (>= 1.0.2), roxygen2 (>= 7.3.3),  
spelling (>= 2.3.2), testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Config/testthat/parallel** false

**Config/roxygen2/version** 8.1.0

**Imports** tools

**NeedsCompilation** no

**Author** Jordan Mark Barbone [aut, cph, cre] (ORCID:  
<<https://orcid.org/0000-0001-9788-3628>>)

**Maintainer** Jordan Mark Barbone <jmbarbone@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-10-07 22:10:09 UTC

## Contents

|                               |   |
|-------------------------------|---|
| cnd_create_registry . . . . . | 2 |
| cnd_document . . . . .        | 3 |

|                             |   |
|-----------------------------|---|
| cnd_exports . . . . .       | 4 |
| cnd_is . . . . .            | 5 |
| condition . . . . .         | 6 |
| format-conditions . . . . . | 9 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>10</b> |
|--------------|-----------|

---

|                     |                              |
|---------------------|------------------------------|
| cnd_create_registry | <i>Create a registration</i> |
|---------------------|------------------------------|

---

**Description**

This function will create a new object with the name as name in the environment where it is called. This is intended to be your package environment, but could potentially be anywhere you want. If an object which is not a cnd:registry object is found with the same name, an error will be thrown.

**Usage**

```
cnd_create_registry(  
  registry = get_package(),  
  overwrite = FALSE,  
  name = ".__CND_REGISTRY__",  
  env = parent.frame()  
)
```

**Arguments**

|           |                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------|
| registry  | The name of the registry                                                                                  |
| overwrite | When TRUE will overwrite                                                                                  |
| name      | The name of the registry variable. Default is intended to prevent potential conflicts with other objects. |
| env       | The environment to assign the registry to                                                                 |

**Details**

Crate a new cnd:registry to the current environment

**Value**

a cnd:registry object, invisibly

**Examples**

```
# In most cases, just having the function in your R/ scripts is good enough,  
# and you can use `cnd_create_registry()` with its defaults. The following  
# examples are for demonstration purposes:  
e <- new.env()  
cnd_create_registry("EXAMPLE", env = e)  
cnd_create_registry("EXAMPLE", overwrite = TRUE)
```

---

|              |                                 |
|--------------|---------------------------------|
| cnd_document | <i>Document your conditions</i> |
|--------------|---------------------------------|

---

## Description

Documents your `conditions()`

## Usage

```
cnd_document(
  package = get_package(),
  registry = package,
  file = file.path("R", paste0(package, "-cnd-conditions.R")),
  cleanup = TRUE
)

cnd_section(fun)
```

## Arguments

|          |                                                                                                                                                                                                                                                                                                  |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| package  | The package to document                                                                                                                                                                                                                                                                          |
| registry | The name of the registry                                                                                                                                                                                                                                                                         |
| file     | The file to save the documentation. This can be a file path, a connection object, or NULL. When file is a path, the directory of the path is searched for files containing # % Generated by cnd: do not edit by hand. These are removed if they are not the same as the generated documentation. |
| cleanup  | If FALSE will not remove files containing # % Generated by cnd: do not edit by hand                                                                                                                                                                                                              |
| fun      | The name of a function                                                                                                                                                                                                                                                                           |

## Value

- `cnd_document()` Conditional on the file argument:
  - when file is a connection, the connection object
  - when file is a path, the path
  - when file is NULL, a character vector of the documentation
  - if no conditions are found, a warning is thrown and NULL is returned
- `cnd_section()` A character vector of the documentation

## conditions

Conditions are generated through the `{cnd}` package. The following conditions are associated with this function:

`cnd::cnd_document_conditions/warning` Documentation will fail when no conditions are found. You may be executing `cnd_document()` too early, before conditions have been registered. You can try to find your conditions with `conditions()`.

`cnd::cnd_document_file/error` The `file` argument to `cnd_document()` must be a file path, a connection object, or NULL to return the documentation as a character vector. The default value should be suitable for standard use cases.

`cnd::cnd_document_pkg_reg/error` Both package and registry must be set to document conditions. You can set a registry by adding `cnd_create_registry()` calls to your package code.

`cnd::cnd_generated_cleanup/message` Some files created during the documentation process may become obsolete while updating your conditions.

`cnd::cnd_generated_write/condition` This condition is signaled when `cnd_document()` needs to write new documentation files.

For more conditions, see: [cnd-cnd-conditions](#)

## Examples

```
file <- file()
cnd_document("cnd", file = file)
readLines(file)

cnd_section("cnd")
```

---

cnd\_exports

*Add conditions to functions*

---

## Description

`[cnd::cnd_exports()]` should be used within a package's building environment.

## Usage

```
cnd_exports(env = parent.frame())
```

## Arguments

`env`                      The package environment

## Value

Nothing, called for its side-effects

## Examples

```
e <- new.env()
registry <- cnd_create_registry("EXAMPLE", env = e)
local(envir = e, {
  my_fun <- function() NULL
  condition(
    "my_condition",
    package = "example_package",
```

```

      exports = "my_fun",
      registry = registry
    )
    cnd_exports()
  })

# conditions are now added to my_fun():
e$my_fun
conditions(e$my_fun)

```

cnd\_is

*is functions for [cnd](#)***Description**

is functions for [cnd](#)

**Usage**

```

is_condition(x)

is_cnd_condition(x)

is_cnd_generator(x, type = c("error", "warning", "message", "condition"))

is_conditioned_function(x)

```

**Arguments**

|      |                          |
|------|--------------------------|
| x    | An object                |
| type | A specific type to check |

**Value**

TRUE or FALSE for the test

**Examples**

```

is_condition(simpleCondition(""))
is_cnd_condition(simpleCondition(""))

con <- condition("is")
is_condition(con)
is_cnd_condition(con)

is_condition(con())
is_cnd_condition(con())

is_cnd_generator(con)

```

```
is_conditioned_function(cnd)
```

---

```
condition
```

```
Conditions
```

---

## Description

`condition()` is used to create a new condition function that itself returns a new condition.

`conditions()` retrieves all conditions based on search values. The parameters serve as filtering arguments.

## Usage

```
condition(
  name,
  message = NULL,
  type = c("condition", "message", "warning", "error"),
  package = get_package(),
  exports = NULL,
  help = NULL,
  registry = package,
  register = !is.null(registry),
  classes = NULL,
  class
)

conditions(
  ...,
  class = NULL,
  type = NULL,
  package = NULL,
  registry = NULL,
  fun = NULL
)

cond(x)

cnd(condition)

conditions(x, ...) <- value

## S3 replacement method for class '`function`'
conditions(x, append = FALSE, ...) <- value

## S3 replacement method for class '`cnd::condition_progenitor`'
conditions(x, ...) <- value
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name      | The name of the new class                                                                                                                                                                                                                                                                                        |
| message   | The message to be displayed when the condition is called. When entered as a character vector, the message is collapsed into a single string. Use explicit line returns to generate new lines in output messages. When a function is used and a character vector returned, each element is treated as a new line. |
| type      | The type of condition: error, warning, or message                                                                                                                                                                                                                                                                |
| package   | The package to which the condition belongs                                                                                                                                                                                                                                                                       |
| exports   | The exported functions to be displayed when the condition is called                                                                                                                                                                                                                                              |
| help      | The help message to be displayed for the condition function                                                                                                                                                                                                                                                      |
| registry  | The name of the registry to store the condition                                                                                                                                                                                                                                                                  |
| register  | Controls registration checks                                                                                                                                                                                                                                                                                     |
| classes   | Additional classes to add to the condition                                                                                                                                                                                                                                                                       |
| class     | <b>Deprecated;</b> use name instead                                                                                                                                                                                                                                                                              |
| ...       | Additional arguments passed to methods                                                                                                                                                                                                                                                                           |
| fun       | if a function is passed, then retrieves the "conditions" attribute                                                                                                                                                                                                                                               |
| x         | An object                                                                                                                                                                                                                                                                                                        |
| condition | A <a href="#">condition_generator</a> object                                                                                                                                                                                                                                                                     |
| value     | A condition                                                                                                                                                                                                                                                                                                      |
| append    | If TRUE, adds to the <b>conditions</b> attribute                                                                                                                                                                                                                                                                 |

**Details**

Conditions

**Value**

- [condition\(\)](#) a [condition\\_generator](#) object
- [conditions\(\)](#) a list of [condition\\_generator](#) objects
- [cond\(\)](#) A [condition\\_generator](#) object
- [cnd\(\)](#) is a wrapper for calling [base::stop\(\)](#), [base::warning\(\)](#), or [base::message\(\)](#); when condition is a type, an error is thrown, and likewise for the other types. When an error isn't thrown, the condition is returned, invisibly.

**[condition\\_generator](#)**

A [condition\\_generator](#) is an object (a special [function](#)) which can be used to create generate a new condition, based on specifications applied in [condition\(\)](#). These functions use ... to absorb extra arguments and contain a special .call parameter. By default, .call captures the parent call from where the [condition\\_generator](#) was created, but users may pass their own call to override this. See [call.](#) in [base::conditionCall\(\)](#)

**condition() conditions**

Conditions are generated through the `{cnd}` package. The following conditions are associated with this function:

`cnd::condition_as_character_error/error` You cannot coerce a `condition_generator` object to a character. This may have occurred when trying to put a condition function through `base::stop()` or `base::warning()`. Instead, call the function first, then pass the result to `base::stop()` or `base::warning()`.

For example:

```
# Instead of this
stop(my_condition)
```

```
# Do this
stop(my_condition())
```

`cnd::condition_message_error/error` Conditions messages are displayed when invoked through `base::conditionMessage()`. You can set a static message by passing through a character vector, or a dynamic message by passing through a function. The function should return a character vector.

When message is not set, a default "there was an error" message is used.

`cnd::condition_message_generator_error/error` `condition_generator` objects are not conditions. You may have made this mistake:

```
x <- condition("my_condition")
conditionMessage(x)
```

Condition generators need to be called first before they can be used as conditions. Try this instead:

```
x <- condition("my_condition")
conditionMessage(x())
```

`cnd::condition_overwrite_warning/warning` Defining a new condition with the same class and package as an existing condition will overwrite the previous definition. It is recommended to either avoid this by fully defining your condition, or creating a new condition instead.

`cnd::invalid_condition_error/error` The class, exports, and help parameters must be a single character string. If you are passing a function, it must be a valid function.

`cnd::match_arg/error` Mostly `base::match.arg()` but with a custom condition

`cnd::no_package_exports_warning/warning` The exports parameter requires a package

For more conditions, see: [cnd-cnd-conditions](#)

**cnd() conditions**

Conditions are generated through the `{cnd}` package. The following conditions are associated with this function:

`cnd::cnd_class_error/error` `cnd()` simple calls the appropriate function: `base::stop()`, `base::warning()`, or `base::message()` based on the type parameter from `condition()`.

For more conditions, see: [cnd-cnd-conditions](#)

**See Also**[cnd-package](#)**Examples**

```
# create a new condition:
cond_bad_value <- condition("bad_value", type = "error")

# use the condition
try(stop(cond_bad_value()))
try(cnd(cond_bad_value()))

# dynamic messages:
cond_class_error <- condition(
  "class_error",
  message = function(x) paste("class cannot be", toString(class(x))),
  type = "error"
)
try(stop(cond_class_error(list())))
```

---

|                   |                          |
|-------------------|--------------------------|
| format-conditions | <i>Format conditions</i> |
|-------------------|--------------------------|

---

**Description**

Formats [condition](#) objects

**Usage**

```
## S3 method for class '`cnd::condition`'
format(x, ..., cli = getOption("cnd.cli.override"))

## S3 method for class '`cnd::condition_generator`'
format(x, ..., cli = getOption("cnd.cli.override"))
```

**Arguments**

|     |                                                                                                                                                                                        |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x   | A <a href="#">condition</a> object                                                                                                                                                     |
| ... | Not used                                                                                                                                                                               |
| cli | If TRUE will use formatting from <a href="#">cli</a> . Default uses an option, "cnd.cli.override", if available, otherwise checks that cli is installed and ansi colors are available. |

**Value**

A character vector

**Examples**

```
format(condition("foo"))
```

# Index

`{cnd}`, 3, 8

`base::conditionCall()`, 7

`base::conditionMessage()`, 8

`base::match.arg()`, 8

`base::message()`, 7, 8

`base::stop()`, 7, 8

`base::warning()`, 7, 8

`cli`, 9

`cnd`, 5

`cnd(condition)`, 6

`cnd()`, 7, 8

`cnd-cnd-conditions`, 4, 8

`cnd-package`, 9

`cnd::cnd_class_error/error`, 8

`cnd::cnd_document_conditions/warning`, 3

`cnd::cnd_document_file/error`, 4

`cnd::cnd_document_pkg_reg/error`, 4

`cnd::cnd_generated_cleanup/message`, 4

`cnd::cnd_generated_write/condition`, 4

`cnd::condition_as_character_error/error`, 8

`cnd::condition_generator(condition)`, 6

`cnd::condition_message_error/error`, 8

`cnd::condition_message_generator_error/error`, 8

`cnd::condition_overwrite_warning/warning`, 8

`cnd::condition_progenitor(condition)`, 6

`cnd::invalid_condition_error/error`, 8

`cnd::match_arg/error`, 8

`cnd::no_package_exports_warning/warning`, 8

`cnd_create_registry`, 2

`cnd_create_registry()`, 4

`cnd_document`, 3

`cnd_document()`, 3, 4

`cnd_exports`, 4

`cnd_is`, 5

`cnd_section(cnd_document)`, 3

`cnd_section()`, 3

`cond(condition)`, 6

`cond()`, 7

`condition`, 6, 9

`condition()`, 6–8

`condition_generator`, 7, 8

`condition_generator(condition)`, 6

`condition_progenitor(condition)`, 6

`conditions(condition)`, 6

`conditions()`, 3, 6, 7

`conditions<- (condition)`, 6

`format-conditions`, 9

`format.cnd::condition`  
(`format-conditions`), 9

`format.cnd::condition_generator`  
(`format-conditions`), 9

`function`, 7

`is_cnd_condition(cnd_is)`, 5

`is_cnd_generator(cnd_is)`, 5

`is_condition(cnd_is)`, 5

`is_conditioned_function(cnd_is)`, 5