Package ‘zephyr’

January 17, 2025

Title Structured Messages and Options

Version 0.1.0

Description Provides a structured framework for consistent user
communication and configuration management for package developers.

License Apache License (>= 2)

URL `https://novonordisk-opensource.github.io/zephyr/`,
`https://github.com/novonordisk-opensource/zephyr`

BugReports `https://github.com/novonordisk-opensource/zephyr/issues`

Depends R (>= 4.1)

Imports cli, glue, rlang (>= 1.0.0)

Suggests callr, checkmate, devtools, knitr, rmarkdown, testthat (>=
3.1.5), usethis, withr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Aksel Thomsen [aut, cre],
Mathias Lerbech Jeppesen [aut],
Cervan Girard [aut],
Kristian Troejelsgaard [aut],
Lovemore Gakava [aut],
Steffen Falgreen Larsen [aut],
Vladimir Obucina [aut],
Novo Nordisk A/S [cph]

Maintainer Aksel Thomsen <`oath@novonordisk.com`>

Repository CRAN

Date/Publication 2025-01-17 09:10:02 UTC

Contents

`create_option` *Create package option*

Description

Use inside your package to setup a `zephyr_option` that you can use in your functions with `get_option()`. The specification is stored inside the environment of your package.

For more information and how to get started see `use_zephyr()`.

Usage

```
create_option(name, default, description = name, .envir = parent.frame())
```

Arguments

`name`	`[character(1)]` Name of the option
`default`	`[any]` Default value of the option
`description`	`[character(1)]` Description of the option
`.envir`	Environment in which the option is defined. Default is suitable for use inside your package.

Value

Invisible `zephyr_option` object

Examples

```
create_option(
  name = "answer",
  default = 42,
  description = "This is supposed to be the question"
)
```

get_all_verbosity_levels — *Get all verbosity levels*

Description

Retrieves all active verbosity levels set for loaded packages.

See also verbosity_level and `get_verbosity_level()`.

Usage

```
get_all_verbosity_levels()
```

Value

Named `[character()]` vector with package as names and their verbosity levels as values.

Examples

```
get_all_verbosity_levels()
```

get_option — *Get value of package option*

Description

Retrieves the value of an `zephyr_option`. The value is looked up in the following order:

1. User defined option: `{pkgname}.{name}`
2. System variable: `R_{PKGNAME}_{NAME}`
3. Default value defined with `create_option()`

And returns the first set value.

Usage

```
get_option(name, .envir = sys.function(which = -1))
```

Arguments

`name`	`[character(1)]` Name of the option
`.envir`	Environment in which the option is defined. Default is suitable for use inside your package.

Details

Environment variables are always defined as character strings. In order to return consistent values the following conversions are applied:

1. If they contain a ";" they are split into a vector using ";" as the delimiter.
2. If the class of the default value is not character, the value is converted to the same class using the naive `as.{class}` function. E.g. conversions to numeric are done with `as.numeric()`.

These conversions are simple in nature and will not cover advanced cases, but we should try to keep our options this simple.

Value

Value of the option

Examples

```
# Retrieve the verbosity level option set by default in zephyr:
get_option(name = "verbosity_level", .envir = "zephyr")
```

`get_verbosity_level` *Get verbosity level*

Description

This function retrieves the `verbosity_level` for your environment using the priority hierarchy as described in verbosity_level.

While the examples use `zephyr`, this function works with any package, and inside a package it is not necessary to specify it; the default value of `.envir` is enough.

It is normally not relevant to query the `verbosity_level` yourself. Instead use the appropriate msg function.

Usage

```
get_verbosity_level(.envir = sys.function(which = -1))
```

Arguments

`.envir`	Environment in which the options are defined. Default is suitable for use inside your package.

Value

`[character(1)]` representing the verbosity level.

Examples

```
# Get the verbosity level
# Note: Specifying the environment is not needed when used inside a package
get_verbosity_level("zephyr")

# Temporarily change verbosity level using an environment variable
withr::with_envvar(
  new = c("R_ZEPHYR_VERBOSITY_LEVEL" = "quiet"),
  code = get_verbosity_level("zephyr")
)

# Temporarily change verbosity level using an option value
withr::with_options(
  new = c("zephyr.verbosity_level" = "minimal"),
  code = get_verbosity_level("zephyr")
)
```

`list_options` *List package options*

Description

List all `zephyr_options` specified in a package. Either as an `list` or as as `character` vector formatted for use in your package documentation.

To document your options use `use_zephyr()` to set everything up, and edit the created template as necessary.

Usage

```
list_options(
  as = c("list", "params", "markdown"),
  .envir = sys.function(which = -1)
)
```

Arguments

`as`	`[character(1)]` Format in which to return the options: • `"list"`: Return a nested list, where each top level element is a list with the specification of an option. • `"params"`: Return a character vector with the `"@param"` tag entries for each option similar to how function parameters are documented with roxygen2. • `"markdown"`: Return a character string with markdown formatted entries for each option.
`.envir`	Environment in which the options are defined. Default is suitable for use inside your package.

Value

`list` or `character` depending on `as`

Examples

```
# List all options in zephyr
x <- list_options(.envir = "zephyr")
print(x)
str(x)

# Create @params tag entries for each option
list_options(as = "params", .envir = "zephyr") |>
  cat()

# List options in markdown format
list_options(as = "markdown", .envir = "zephyr") |>
  cat()
```

`msg` *Write messages based on verbosity level*

Description

The `msg()` function is a general utility function for writing messages to the console based on the verbosity_level set for your session and package.

For simple messages in your functions the recommended approach is to use the following wrappers for consistency across packages:

- `msg_success()`: To indicate a successful operation. Wrapper around `msg()` using `cli::cli_alert_success()` to display the message.
- `msg_danger()`: To indicate a failed operation. Wrapper around `msg()` using `cli::cli_alert_danger()` to display the message.
- `msg_warning()`: To indicate a warning. Wrapper around `msg_verbose()` using `cli::cli_alert_warning()` to display the message.
- `msg_info()`: To provide additional information. Wrapper around `msg_verbose()` using `cli::cli_alert_info()` to display the message.

For more control of how the messages are displayed use:

- `msg()`: To write messages using custom `msg_fun` functions and define your own verbosity levels to write.
- `msg_verbose()`: To write verbose messages with a custom `msg_fun`.
- `msg_debug()`: To to report messages only relevant when debugging.

For more information on the verbosity levels, see verbosity_level.

Usage

```
msg(
  message,
  levels_to_write = c("minimal", "verbose", "debug"),
  msg_fun = cli::cli_alert,
  ...,
  .envir = parent.frame()
)

msg_verbose(message, msg_fun = cli::cli_alert, ..., .envir = parent.frame())

msg_debug(message, msg_fun = cli::cli_alert, ..., .envir = parent.frame())

msg_success(message, ..., .envir = parent.frame())

msg_danger(message, ..., .envir = parent.frame())

msg_warning(message, ..., .envir = parent.frame())

msg_info(message, ..., .envir = parent.frame())
```

Arguments

`message`	`character` string with the text to display.
`levels_to_write`	`character` vector with the verbosity levels for which the message should be displayed. Options are `minimal`, `verbose`, and `debug`.
`msg_fun`	The function to use for writing the message. Most commonly from the cli package. Default is `cli::cli_alert()`.
`...`	Additional arguments to pass to `msg_fun()`
`.envir`	The `environment` to use for evaluating the verbosity level. Default `parent.frame()` will be sufficient for most use cases. Parsed on to `msg_fun()`.

Value

Return from `msg_fun()`

Examples

```
msg("General message")
msg_success("Operation successful")
msg_danger("Operation failed")
msg_warning("Warning message")
msg_info("Additional information")
```

report_checkmate_assertions

Report collection of assertions

Description

Improved reporting of an `AssertCollection` created with the `checkmate::makeAssertCollection()` using `cli::cli_abort()` instead of `checkmate::reportAssertions()` in order to provide a more informative error message.

The function is intended to be used inside a function that performs assertions on its input arguments.

Usage

```
report_checkmate_assertions(
  collection,
  message = "Invalid input(s):",
  .envir = parent.frame()
)
```

Arguments

`collection`	`[AssertCollection]` A collection of assertions created with `checkmate::makeAssertCollection()`.
`message`	`[character(1)]` string with the header of the error message if any assertions failed
`.envir`	The `[environment]` to use for the error message. Default `parent.frame()` will be sufficient for most use cases.

Value

`invisible(TRUE)`

Examples

```
add_numbers <- function(a, b) {
  collection <- checkmate::makeAssertCollection()
  checkmate::assert_numeric(x = a, add = collection)
  checkmate::assert_numeric(x = b, add = collection)
  report_checkmate_assertions(collection)
  return(a + b)
}

add_numbers(1, 2)
try(add_numbers(1, "b"))
try(add_numbers("a", "b"))
```

`use_zephyr` *Use zephyr options and verbosity levels*

Description

Utility function to set up the use of zephyr options and verbosity_level in your package.

Creates the file `R/{pkgname}-options.R` with boiler plate code to setup and document options.

This code also creates an package specific `verbosity_level` option, enabling you to control the verbosity of your package functions using the msg functions.

Usage

```
use_zephyr()
```

Value

```
invisible(TRUE)
```

Examples

```
use_zephyr()
```

`verbosity_level` *Verbosity level to control package behavior*

Description

In zephyr we define a central verbosity level to control the amount of messages the user receives when using zephyr and other packages in the ecosystem.

Verbosity level can be any of the four values below:

1. `quiet`: No messages are displayed.
2. `minimal`: Only essential messages are displayed.
3. `verbose` (*default*): More informative messages are displayed.
4. `debug`: Detailed messages for debugging are displayed.

See `use_zephyr()` and msg for how to implement the use of verbosity levels in your package and its functions.

Verbosity level is a special kind of option that can be scoped both for a specific package and and globally for the ecosystem (assigned to the zephyr package). It can be set using either R `options()` or environment variables.

Verbosity level is retrieved using the `get_verbosity_level()` function. Since the level can have multiples scopes, the following hierarchy is used:

1. Package specific option: `{pkgname}.verbosity_level`
2. Package specific environment variable: `R_{PKGNAME}_VERBOSITY_LEVEL`
3. Ecosystem wide option: `zephyr.verbosity_level`
4. Ecosystem wide environment variable (`R_ZEPHYR_VERBOSITY_LEVEL`)
5. Default value specified in zephyr (`verbose`, see above).

In order to see all registered verbosity levels across scopes call `get_all_verbosity_levels()`.

Examples

```
get_verbosity_level("zephyr")
get_all_verbosity_levels()
```

Index