Package ‘hera’

May 8, 2026

Title Companion to the 'xeus-r' 'jupyter' Kernel

Version 0.1.1

Description Set of R functions to be coupled with the 'xeus-r' 'jupyter' kernel in order to drive execution of code in notebook input cells, how R objects are to be displayed in output cells, and handle two way communication with the front end through comms.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports cli, evaluate, glue, IRdisplay, jsonlite, R6, repr, rlang, tools, utils

URL https://github.com/jupyter-xeus/xeus-r

BugReports https://github.com/jupyter-xeus/xeus-r/issues

NeedsCompilation no

Author Romain François [aut, cre], QuantStack [cph]

Maintainer Romain François <romain@tada.science>

Repository CRAN

Date/Publication 2025-04-10 09:10:05 UTC

Contents

cell_options 2
clear_output 2
Comm 3
CommManagerClass 5
complete 7
display_data 8
is_xeusr 8
mime_bundle 9
mime_types 9
View 10

Index 11

`cell_options` *Options for current jupyter cell*

Description

Options for current jupyter cell

Usage

```
cell_options(...)
```

Arguments

`...`	options to set locally to the notebook cell. Forwarded to `rlang::local_options()`.

Examples

```
## Not run:
  cell_options(repr.plot.bg = "gray")

## End(Not run)
```

`clear_output` *Clear output*

Description

Clear output

Usage

```
clear_output(wait = FALSE)
```

Arguments

`wait`	Should this wait

Value

NULL invisibly

Examples

```
## Not run:
  clear_output()

## End(Not run)
```

Comm *Comm class*

Description

Comm class

Comm class

Active bindings

`id` id of the comm. read only.

`target_name` name of the target for this comm

Methods

Public methods:

- `Comm$new()`
- `Comm$open()`
- `Comm$close()`
- `Comm$send()`
- `Comm$on_close()`
- `Comm$on_message()`
- `Comm$print()`
- `Comm$clone()`

Method `new()`:

Usage:

```
Comm$new(xp, description = "")
```

Arguments:

`xp` external pointer to an instance of the C++ class 'xeus::xcomm'

`description` description of this comm

Method `open()`:

Usage:

```
Comm$open(data = NULL, metadata = NULL)
```

Arguments:

`data` data

`metadata` metadata

Method `close()`**:**

Usage:

```
Comm$close(data = NULL, metadata = NULL)
```

Arguments:

`data` data

`metadata` metadata

Method `send()`**:**

Usage:

```
Comm$send(data = NULL, metadata = NULL)
```

Arguments:

`data` data

`metadata` metadata

Method `on_close()`**:**

Usage:

```
Comm$on_close(handler)
```

Arguments:

`handler` function to call when the comm is closed

Method `on_message()`**:**

Usage:

```
Comm$on_message(handler)
```

Arguments:

`handler` function to call when receiving a message

Method `print()`**:**

Usage:

```
Comm$print()
```

Returns: information about the comm

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

Usage:

```
Comm$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

CommManagerClass *Comm Manager class*

Description

Instance of the CommManagerClass class.

Usage

```
CommManager
```

Format

An object of class `CommManagerClass` (inherits from `R6`) of length 11.

Methods

Public methods:

- `CommManagerClass$new()`
- `CommManagerClass$register_comm_target()`
- `CommManagerClass$unregister_comm_target()`
- `CommManagerClass$new_comm()`
- `CommManagerClass$comms()`
- `CommManagerClass$target_callback()`
- `CommManagerClass$preserve()`
- `CommManagerClass$release()`
- `CommManagerClass$get_comm_info()`
- `CommManagerClass$clone()`

Method `new()`:

Usage:

```
CommManagerClass$new(..., error_call = caller_env())
```

Arguments:

`...` currently unused

`error_call` see `rlang::args_error_context()`

Method `register_comm_target()`:

Usage:

```
CommManagerClass$register_comm_target(
  target_name,
  callback = function(comm, message) {
 }
)
```

Arguments:

`target_name` name of the comm target, e.g. "jupyter.widgets"
`callback` callback function taking two arguments 'comm' and 'message'.

Method `unregister_comm_target()`**:**

Usage:

```
CommManagerClass$unregister_comm_target(target_name)
```

Arguments:

`target_name` name of the comm target

Method `new_comm()`**:**

Usage:

```
CommManagerClass$new_comm(target_name, description = "")
```

Arguments:

`target_name` name of the target
`description` description of the comm

Method `comms()`**:**

Usage:

```
CommManagerClass$comms()
```

Returns: the list of currently open comms

Method `target_callback()`**:**

Usage:

```
CommManagerClass$target_callback(target_name)
```

Arguments:

`target_name` name of the comm target

Returns: the callback for that target name

Method `preserve()`**:**

Usage:

```
CommManagerClass$preserve(comm)
```

Arguments:

`comm` Comm instance to preserve

Method `release()`**:**

Usage:

```
CommManagerClass$release(comm)
```

Arguments:

`comm` Comm instance to release

Method `get_comm_info()`**:**

Usage:

```
CommManagerClass$get_comm_info(target_name = NULL)
```

Arguments:

`target_name` name of the target to get info about all comm opens. If NULL, info for comms for all targets are returned.

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

Usage:

```
CommManagerClass$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`complete` *Code completion*

Description

Code completion

Usage

```
complete(code, cursor_pos = nchar(code))
```

Arguments

`code`	R code to complete
`cursor_pos`	position of the cursor

Value

a list that contains potential completions as the first item

Examples

```
complete("rnorm(")
```

display_data *Display data*

Description

Display data

Usage

```
display_data(data = NULL, metadata = NULL)
```

Arguments

data	data to display
metadata	potential metadata

Examples

```
## Not run:
  display_data(mtcars)

## End(Not run)
```

is_xeusr *Is this a running xeusr jupyter kernel*

Description

Is this a running xeusr jupyter kernel

Usage

```
is_xeusr()
```

Value

TRUE if the current session is running in a xeusr kernel

Examples

```
is_xeusr()
```

mime_bundle *bundle an object*

Description

bundle an object

Usage

```
mime_bundle(x, mimetypes = mime_types(x), ...)
```

Arguments

x	an object
mimetypes	mime types
...	extra currently unused parameters

See Also

IRdisplay::prepare_mimebundle which this currently wraps around

Examples

```
mime_bundle(letters)
```

mime_types *MIME types supported by an object*

Description

MIME types supported by an object

Usage

```
mime_types(x)
```

Arguments

x	an object

Value

a character vector of its supported mime types

Examples

```
mime_types(letters)
mime_types(mtcars)
```

`View` *View*

Description

View

Usage

```
View(x, title)
```

Arguments

`x`	something to display
`title`	title of the display

Examples

```
## Not run:
  View(mtcars)

## End(Not run)
```

Index

∗ **datasets**
    CommManagerClass, 5

cell_options, 2
clear_output, 2
Comm, 3, *6*
CommManager *(*CommManagerClass*)*, 5
CommManagerClass, *5*, 5
complete, 7

display_data, 8

is_xeusr, 8

mime_bundle, 9
mime_types, 9

rlang::args_error_context(), *5*
rlang::local_options(), *2*

View, 10