Package 'opalr'

November 3, 2020

Version 1.5.0

Title 'Opal' Data Repository Client and 'DataSHIELD' Utils

Depends R (>= 3.1), httr

Imports jsonlite, mime, progress

Suggests e1071, knitr, knitrBootstrap, rmarkdown, tibble, testthat

Description Data integration Web application for biobanks by 'OBiBa'. 'Opal' is the core database application for biobanks. Participant data, once collected from any data source, must be integrated and stored in a central data repository under a uniform model. 'Opal' is such a central repository. It can import, process, validate, query, analyze, report, and export data. 'Opal' is typically used in a research center to analyze the data acquired at assessment centres. Its ultimate purpose is to achieve seamless data-sharing among biobanks. This 'Opal' client allows to interact with 'Opal' web services and to perform operations on the R server side. 'DataSHIELD' administration tools are also provided.

License GPL-3

URL https://www.obiba.org/ https://www.obiba.org/pages/products/opal/ https://doi.org/10.1093/ije/dyx180 http://www.datashield.ac.uk/

BugReports https://github.com/obiba/opalr

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Yannick Marcon [aut, cre] (<https://orcid.org/0000-0003-0138-2023>), Amadou Gaye [ctb] (<https://orcid.org/0000-0002-1180-2792>), OBiBa group [cph]

Maintainer Yannick Marcon <yannick.marcon@obiba.org>

Repository CRAN

Date/Publication 2020-11-03 10:20:03 UTC

R topics documented:

dictionary.annotate . . . 5
dictionary.annotate.harmo_status . . . 6
dictionary.annotations . . . 6
dictionary.apply . . . 7
dsadmin.get_method . . . 8
dsadmin.get_methods . . . 9
dsadmin.get_options . . . 10
dsadmin.installed_package . . . 10
dsadmin.install_github_package . . . 11
dsadmin.install_local_package . . . 12
dsadmin.install_package . . . 13
dsadmin.package_description . . . 14
dsadmin.package_descriptions . . . 15
dsadmin.perm . . . 16
dsadmin.perm_add . . . 16
dsadmin.perm_delete . . . 17
dsadmin.remove_package . . . 18
dsadmin.rm_method . . . 18
dsadmin.rm_methods . . . 19
dsadmin.rm_option . . . 20
dsadmin.rm_package_methods . . . 21
dsadmin.set_method . . . 21
dsadmin.set_option . . . 22
dsadmin.set_package_methods . . . 23
harmo.annotate . . . 24
harmo.annotate.status . . . 25
harmo.annotations . . . 25
harmo.dictionary_apply . . . 26
harmo.dictionary_update . . . 27
harmo.table_get . . . 28
harmo.table_save . . . 29
oadmin.installed_devtools . . . 30
oadmin.installed_package . . . 31
oadmin.installed_packages . . . 32
oadmin.install_bioconductor_package . . . 32
oadmin.install_devtools . . . 33
oadmin.install_github . . . 34
oadmin.install_github_package . . . 35
oadmin.install_local_package . . . 36
oadmin.install_package . . . 36
oadmin.package_description . . . 37
oadmin.perm . . . 38
oadmin.perm_add . . . 39
oadmin.perm_delete . . . 39
oadmin.remove_package . . . 40
opal.annotate . . . 41

opal.annotations . . . 42
opal.assign . . . 43
opal.assign.data . . . 44
opal.assign.resource . . . 45
opal.assign.script . . . 46
opal.assign.table . . . 46
opal.assign.table.tibble . . . 48
opal.as_md_table . . . 49
opal.attribute_values . . . 50
opal.command . . . 51
opal.commands . . . 52
opal.commands_rm . . . 52
opal.command_result . . . 53
opal.command_rm . . . 54
opal.datasource . . . 54
opal.datasources . . . 55
opal.delete . . . 56
opal.execute . . . 56
opal.execute.source . . . 57
opal.file . . . 58
opal.file_cp . . . 59
opal.file_download . . . 59
opal.file_ls . . . 60
opal.file_mkdir . . . 61
opal.file_mkdir_tmp . . . 62
opal.file_mv . . . 62
opal.file_read . . . 63
opal.file_rm . . . 64
opal.file_upload . . . 65
opal.file_write . . . 65
opal.get . . . 66
opal.load_package . . . 67
opal.login . . . 67
opal.logout . . . 69
opal.post . . . 69
opal.project . . . 70
opal.projects . . . 71
opal.project_create . . . 72
opal.project_delete . . . 73
opal.project_exists . . . 73
opal.project_perm . . . 74
opal.project_perm_add . . . 75
opal.project_perm_delete . . . 76
opal.put . . . 76
opal.report . . . 77
opal.resource . . . 78
opal.resources . . . 79
opal.resources_perm . . . 79

opal.resources_perm_add . . . 80
opal.resources_perm_delete . . . 81
opal.resource_create . . . 81
opal.resource_delete . . . 83
opal.resource_exists . . . 83
opal.resource_get . . . 84
opal.resource_perm . . . 85
opal.resource_perm_add . . . 86
opal.resource_perm_delete . . . 87
opal.rm . . . 87
opal.symbols . . . 88
opal.symbol_import . . . 89
opal.symbol_rm . . . 90
opal.symbol_save . . . 90
opal.table . . . 91
opal.tables . . . 92
opal.tables_perm . . . 92
opal.tables_perm_add . . . 93
opal.tables_perm_delete . . . 94
opal.table_create . . . 94
opal.table_delete . . . 95
opal.table_dictionary_get . . . 96
opal.table_dictionary_update . . . 96
opal.table_exists . . . 97
opal.table_get . . . 98
opal.table_perm . . . 99
opal.table_perm_add . . . 100
opal.table_perm_delete . . . 100
opal.table_save . . . 101
opal.table_truncate . . . 102
opal.task . . . 103
opal.tasks . . . 104
opal.task_cancel . . . 104
opal.task_wait . . . 105
opal.taxonomies . . . 106
opal.taxonomy . . . 106
opal.taxonomy_delete . . . 107
opal.taxonomy_download . . . 108
opal.taxonomy_upload . . . 108
opal.terms . . . 109
opal.unload_package . . . 110
opal.valueset . . . 111
opal.variable . . . 111
opal.variables . . . 112
opal.version_compare . . . 113
opal.vocabularies . . . 114
opal.vocabulary . . . 114
opal.workspaces . . . 115

opal.workspace_rm 116
opal.workspace_save 116

Index **118**

`dictionary.annotate` *Set variable annotation with a taxonomy term*

Description

Apply or remove an annotation from a set of variables.

Usage

```
dictionary.annotate(
  tibble,
  variables = NULL,
  taxonomy = "Mlstr_area",
  vocabulary,
  term
)
```

Arguments

`tibble`	Tibble to be annotated.
`variables`	A character vector of variable names to be annotated. If NULL or empty, all the columns of the tibble will be annotated.
`taxonomy`	The taxonomy to which the vocabulary belongs. If NULL, the annotation is a simple attribute (i.e. without a taxonomy reference).
`vocabulary`	The vocabulary to which the term belongs.
`term`	The term to apply. If NULL, the annotation will be deleted.

Value

The annotated tibble

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
cqx <- dictionary.annotate(cqx,
  variables = c("A_SDC_EDU_LEVEL", "A_SDC_EDU_LEVEL_AGE"),
  taxonomy = "Mlstr_area",
  vocabulary = "Sociodemographic_economic_characteristics",
  term = "Education")
opal.logout(o)

## End(Not run)
```

dictionary.annotate.harmo_status

Set variable annotation with Harmonization Status term

Description

Apply or remove an harmonization status annotation from a set of variables. The harmonization status is described by the "status" vocabulary in the "Mlstr_harmo" taxonomy.

Usage

```
dictionary.annotate.harmo_status(tibble, variables = NULL, status)
```

Arguments

tibble	Tibble to be annotated.
variables	A character vector of variable names to be annotated. If NULL or empty, all the columns of the tibble will be annotated.
status	The harmonization status to apply: 'complete', 'undetermined', 'impossible' or 'na'. If NULL, the annotation will be deleted.

Value

The annotated tibble

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
cqx <- dictionary.annotate.harmo_status(cqx,
  variables = c("A_SDC_EDU_LEVEL", "A_SDC_EDU_LEVEL_AGE"),
  status = "complete")
opal.logout(o)

## End(Not run)
```

dictionary.annotations

List the annotations

Description

List the annotations of each of the variables.

Usage

```
dictionary.annotations(
  tibble,
  variables = NULL,
  taxonomy = NULL,
  vocabulary = NULL
)
```

Arguments

tibble	Tibble to be annotated
variables	A character vector of variable names to be inspected. If NULL or empty, all the columns of the tibble will be inspected.
taxonomy	Filter by taxonomy name(s) (if provided).
vocabulary	Filter by vocabulary name(s) (if provided).

Value

A data frame in long format (one row per annotation).

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
annot <- dictionary.annotations(cqx, taxonomy = "Mlstr_harmo", vocabulary = "status")
opal.logout(o)

## End(Not run)
```

---

dictionary.apply *Apply the dictionary to a tibble*

Description

Apply the dictionary described in a Opal Excel format as attributes of the tibble's columns.

Usage

```
dictionary.apply(tibble, variables, categories = NULL)
```

Arguments

tibble	Tibble to be decorated.
variables	A data frame with one row per variable (column name) and then one column per property/attribute.
categories	A data frame with one row per category (columns variable and name) and then column per property/attribute.

Examples

```
## Not run:
data <- tibble::as_tibble(mtcars)
variables <- tibble::tribble(
  ~name, ~valueType, ~`label:en`,  ~`Namespace::Name`, ~unit, ~repeatable, ~index,
  "mpg", "decimal", "Mpg label",  "Value1", "years", 0, 1,
  "cyl", "decimal", "Cyl label",  "Value2", "kg/m2", 0, 2,
  "disp", "decimal", "Disp label", NA, NA, 1, 3
)
categories <- tibble::tribble(
  ~variable, ~name, ~missing, ~`label:en`, ~`label:fr`,
  "cyl", "4", 0, "Four", "Quatre",
  "cyl", "6", 0, "Six", "Six",
  "cyl", "8", 1, "Height", "Huit"
)
data <- dictionary.apply(data, variables, categories)

## End(Not run)
```

`dsadmin.get_method` *Get a DataSHIELD method*

Description

Get a DataSHIELD method

Usage

```
dsadmin.get_method(opal, name, type = "aggregate")
```

Arguments

`opal`	Opal object or list of opal objects.
`name`	Name of the method, as it is accessed by DataSHIELD users.
`type`	Type of the method: "aggregate" (default) or "assign"

See Also

Other DataSHIELD functions: `dsadmin.get_methods()`, `dsadmin.get_options()`, `dsadmin.install_github_package()`, `dsadmin.install_local_package()`, `dsadmin.install_package()`, `dsadmin.installed_package()`, `dsadmin.package_descriptions()`, `dsadmin.package_description()`, `dsadmin.remove_package()`, `dsadmin.rm_methods()`, `dsadmin.rm_method()`, `dsadmin.rm_option()`, `dsadmin.rm_package_methods()`, `dsadmin.set_method()`, `dsadmin.set_option()`, `dsadmin.set_package_methods()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.get_method(o, 'class')
opal.logout(o)

## End(Not run)
```

`dsadmin.get_methods` *Get DataSHIELD methods*

Description

Get DataSHIELD methods

Usage

```
dsadmin.get_methods(opal, type = "aggregate")
```

Arguments

`opal`	Opal object or list of opal objects.
`type`	Type of the method: "aggregate" (default) or "assign"

See Also

Other DataSHIELD functions: `dsadmin.get_method()`, `dsadmin.get_options()`, `dsadmin.install_github_package()`, `dsadmin.install_local_package()`, `dsadmin.install_package()`, `dsadmin.installed_package()`, `dsadmin.package_descriptions()`, `dsadmin.package_description()`, `dsadmin.remove_package()`, `dsadmin.rm_methods()`, `dsadmin.rm_method()`, `dsadmin.rm_option()`, `dsadmin.rm_package_methods()`, `dsadmin.set_method()`, `dsadmin.set_option()`, `dsadmin.set_package_methods()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.get_methods(o)
opal.logout(o)

## End(Not run)
```

dsadmin.get_options *Get the DataSHIELD options*

Description

Get the DataSHIELD options

Usage

```
dsadmin.get_options(opal)
```

Arguments

opal	Opal object or list of opal objects.

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.get_options(o)
opal.logout(o)

## End(Not run)
```

dsadmin.installed_package *Check DataSHIELD package*

Description

Check if a DataSHIELD package is installed.

Usage

```
dsadmin.installed_package(opal, pkg)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.

Value

TRUE if installed

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.installed_package(o, 'dsBase')
opal.logout(o)

## End(Not run)
```

---

dsadmin.install_github_package

Install a DataSHIELD package from GitHub

Description

Install a package from a DataSHIELD source repository on GitHub.

Usage

```
dsadmin.install_github_package(
  opal,
  pkg,
  username = "datashield",
  ref = "master"
)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
username	GitHub username/organization of the git repository. Default is 'datashield'.
ref	Desired git reference (could be a commit, tag, or branch name). Default is 'master'.

Value

TRUE if installed

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.install_github_package(o, 'dsOmics', username='isglobal-brge')
opal.logout(o)

## End(Not run)
```

---

dsadmin.install_local_package

Install a DataSHIELD package from a local archive file

Description

Install a package from a package archive file, resulting from the build of a server-side DataSHIELD package. This will upload the archive file and run its installation in the R server.

Usage

```
dsadmin.install_local_package(opal, path)
```

Arguments

opal	Opal object or list of opal objects.
path	Path to the package archive, ending with .

Value

TRUE if installed

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# install a pre-built local archive file
dsadmin.install_local_package(o, '~/dsExposome_1.0.0.tar.gz')
# or build archive file from local package source (in current working folder)
dsadmin.install_local_package(o, devtools::build())
opal.logout(o)

## End(Not run)
```

`dsadmin.install_package` *Install a DataSHIELD package*

Description

Install a package from DataSHIELD public package repository or (if Git reference and GitHub username is provided) from DataSHIELD source repository on GitHub.

Usage

```
dsadmin.install_package(opal, pkg, githubusername = NULL, ref = NULL)
```

Arguments

`opal`	Opal object or list of opal objects.
`pkg`	Package name.
`githubusername`	GitHub username of git repository. If NULL (default), try to install from DataSHIELD package repository.
`ref`	Desired git reference (could be a commit, tag, or branch name). If NULL (default), try to install from DataSHIELD package repository.

Value

TRUE if installed

See Also

Other DataSHIELD functions: `dsadmin.get_methods()`, `dsadmin.get_method()`, `dsadmin.get_options()`, `dsadmin.install_github_package()`, `dsadmin.install_local_package()`, `dsadmin.installed_package()`, `dsadmin.package_descriptions()`, `dsadmin.package_description()`, `dsadmin.remove_package()`, `dsadmin.rm_methods()`, `dsadmin.rm_method()`, `dsadmin.rm_option()`, `dsadmin.rm_package_methods()`, `dsadmin.set_method()`, `dsadmin.set_option()`, `dsadmin.set_package_methods()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.install_package(o, 'dsBase')
opal.logout(o)

## End(Not run)
```

---

dsadmin.package_description

Get DataSHIELD package description

Description

Get DataSHIELD package description

Usage

```
dsadmin.package_description(opal, pkg, fields = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
fields	A character vector giving the fields to extract from each package's DESCRIPTION file in addition to the default ones, or NULL (default). Unavailable fields result in NA values.

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.package_description(o, 'dsBase')
opal.logout(o)

## End(Not run)
```

dsadmin.package_descriptions *Get DataSHIELD package descriptions*

Description

Get DataSHIELD package descriptions

Usage

```
dsadmin.package_descriptions(opal, fields = NULL, df = TRUE)
```

Arguments

opal	Opal object or list of opal objects.
fields	A character vector giving the fields to extract from each package's DESCRIPTION file in addition to the default ones, or NULL (default). Unavailable fields result in NA values.
df	Return a data.frame (default is TRUE)

Value

The DataSHIELD package descriptions as a data.frame or a list

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.package_descriptions(o)
opal.logout(o)

## End(Not run)
```

dsadmin.perm *Get the DataSHIELD permissions*

Description

Get the permissions that were applied to the DataSHIELD service.

Usage

```
dsadmin.perm(opal)
```

Arguments

opal	Opal connection object.

Value

A data.frame with columns: subject, type, permission

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.perm_add(o, c('andrei', 'valentina'), 'user', 'use')
dsadmin.perm(o)
dsadmin.perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

dsadmin.perm_add *Add or update a DataSHIELD permission*

Description

Add or update a permission on the DataSHIELD service.

Usage

```
dsadmin.perm_add(opal, subject, type = "user", permission)
```

Arguments

opal	Opal connection object.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.
permission	The permission to apply: use or administrate.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.perm_add(o, c('andrei', 'valentina'), 'user', 'use')
dsadmin.perm(o)
dsadmin.perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

`dsadmin.perm_delete` *Delete a DataSHIELD permission*

Description

Delete a permission that was applied to the DataSHIELD service. Silently returns when there is no such permission.

Usage

```
dsadmin.perm_delete(opal, subject, type = "user")
```

Arguments

`opal`	Opal connection object.
`subject`	A vector of subject identifiers: user names or group names (depending on the type).
`type`	The type of subject: user (default) or group.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.perm_add(o, c('andrei', 'valentina'), 'user', 'use')
dsadmin.perm(o)
dsadmin.perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

dsadmin.remove_package *Remove DataSHIELD package*

Description

Remove a DataSHIELD package permanently.

Usage

```
dsadmin.remove_package(opal, pkg)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.remove_package(o, 'dsBase')
opal.logout(o)

## End(Not run)
```

dsadmin.rm_method *Remove DataSHIELD method*

Description

Remove DataSHIELD method

Usage

```
dsadmin.rm_method(opal, name, type = "aggregate")
```

Arguments

opal	Opal object or list of opal objects.
name	Name of the method, as it is accessed by DataSHIELD users.
type	Type of the method: "aggregate" (default) or "assign"

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.rm_method(o, 'foo')
opal.logout(o)

## End(Not run)
```

---

dsadmin.rm_methods *Remove DataSHIELD methods.*

Description

Remove DataSHIELD methods.

Usage

```
dsadmin.rm_methods(opal, type = NULL)
```

Arguments

opal	Opal object or list of opal objects.
type	Type of the method: "aggregate" or "assign". Default is NULL (=all type of methods).

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.rm_methods(o)
opal.logout(o)

## End(Not run)
```

`dsadmin.rm_option` *Remove a DataSHIELD option*

Description

Remove a DataSHIELD option

Usage

```
dsadmin.rm_option(opal, name)
```

Arguments

`opal`	Opal object or list of opal objects.
`name`	Name of the option

See Also

Other DataSHIELD functions: `dsadmin.get_methods()`, `dsadmin.get_method()`, `dsadmin.get_options()`, `dsadmin.install_github_package()`, `dsadmin.install_local_package()`, `dsadmin.install_package()`, `dsadmin.installed_package()`, `dsadmin.package_descriptions()`, `dsadmin.package_description()`, `dsadmin.remove_package()`, `dsadmin.rm_methods()`, `dsadmin.rm_method()`, `dsadmin.rm_package_methods()`, `dsadmin.set_method()`, `dsadmin.set_option()`, `dsadmin.set_package_methods()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.rm_option(o, 'foo')
opal.logout(o)

## End(Not run)
```

dsadmin.rm_package_methods *Remove DataSHIELD package methods*

Description

Remove DataSHIELD aggregate and assign methods defined by the package.

Usage

```
dsadmin.rm_package_methods(opal, pkg, type = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
type	Type of the method: "aggregate" or "assign". Default is NULL (=all type of methods).

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.rm_package_methods(o, 'dsBase')
opal.logout(o)

## End(Not run)
```

dsadmin.set_method *Set DataSHIELD method*

Description

Set DataSHIELD method

Usage

```
dsadmin.set_method(opal, name, func = NULL, path = NULL, type = "aggregate")
```

Arguments

opal	Opal object or list of opal objects.
name	Name of the method, as it will be accessed by DataSHIELD users.
func	Function name.
path	Path to the R file containing the script (mutually exclusive with func).
type	Type of the method: "aggregate" (default) or "assign"

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_option(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.set_method(o, 'foo', 'base::mean')
opal.logout(o)

## End(Not run)
```

---

dsadmin.set_option *Set DataSHIELD option*

Description

Set a DataSHIELD option (add or update).

Usage

```
dsadmin.set_option(opal, name, value)
```

Arguments

opal	Opal object or list of opal objects.
name	Name of the option
value	Value of the option

See Also

Other DataSHIELD functions: dsadmin.get_methods(), dsadmin.get_method(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_descriptions(), dsadmin.package_description(), dsadmin.remove_package(), dsadmin.rm_methods(), dsadmin.rm_method(), dsadmin.rm_option(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_package_methods()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.set_option(o, 'foo', 'bar')
opal.logout(o)

## End(Not run)
```

`dsadmin.set_package_methods` *Set DataSHIELD package methods*

Description

Declare DataSHIELD aggregate and assign methods as defined by the package.

Usage

```
dsadmin.set_package_methods(opal, pkg, type = NULL)
```

Arguments

`opal`	Opal object or list of opal objects.
`pkg`	Package name.
`type`	Type of the method: "aggregate" or "assign". Default is NULL (=all type of methods).

Value

TRUE if successfull

See Also

Other DataSHIELD functions: `dsadmin.get_methods()`, `dsadmin.get_method()`, `dsadmin.get_options()`, `dsadmin.install_github_package()`, `dsadmin.install_local_package()`, `dsadmin.install_package()`, `dsadmin.installed_package()`, `dsadmin.package_descriptions()`, `dsadmin.package_description()`, `dsadmin.remove_package()`, `dsadmin.rm_methods()`, `dsadmin.rm_method()`, `dsadmin.rm_option()`, `dsadmin.rm_package_methods()`, `dsadmin.set_method()`, `dsadmin.set_option()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.set_package_methods(o, 'dsBase')
opal.logout(o)

## End(Not run)
```

harmo.annotate *Set variable annotation with a taxonomy term (deprecated)*

Description

Deprecated: use dictionary.annotate instead.

Usage

```
harmo.annotate(
  tibble,
  variables = NULL,
  taxonomy = "Mlstr_area",
  vocabulary,
  term
)
```

Arguments

tibble	Tibble to be annotated.
variables	A character vector of variable names to be annotated. If NULL or empty, all the columns of the tibble will be annotated.
taxonomy	The taxonomy to which the vocabulary belongs. If NULL, the annotation is a simple attribute (i.e. without a taxonomy reference).
vocabulary	The vocabulary to which the term belongs.
term	The term to apply. If NULL, the annotation will be deleted.

Value

The annotated tibble

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
cqx <- dictionary.annotate(cqx,
  variables = c("A_SDC_EDU_LEVEL", "A_SDC_EDU_LEVEL_AGE"),
  taxonomy = "Mlstr_area",
  vocabulary = "Sociodemographic_economic_characteristics",
  term = "Education")
opal.logout(o)

## End(Not run)
```

harmo.annotate.status *Set variable annotation with Harmonization Status term (deprecated)*

Description

Deprecated: use dictionary.annotate.harmo_status instead.

Usage

```
harmo.annotate.status(tibble, variables = NULL, status)
```

Arguments

tibble	Tibble to be annotated.
variables	A character vector of variable names to be annotated. If NULL or empty, all the columns of the tibble will be annotated.
status	The harmonization status to apply: 'complete', 'undetermined', 'impossible' or 'na'. If NULL, the annotation will be deleted.

Value

The annotated tibble

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
cqx <- dictionary.annotate.harmo_status(cqx,
  variables = c("A_SDC_EDU_LEVEL", "A_SDC_EDU_LEVEL_AGE"),
  status = "complete")
opal.logout(o)

## End(Not run)
```

harmo.annotations *List the annotations (deprecated)*

Description

Deprecated: use dictionary.annotations instead.

Usage

```
harmo.annotations(tibble, variables = NULL, taxonomy = NULL, vocabulary = NULL)
```

Arguments

tibble	Tibble to be annotated
variables	A character vector of variable names to be inspected. If NULL or empty, all the columns of the tibble will be inspected.
taxonomy	Filter by taxonomy name(s) (if provided).
vocabulary	Filter by vocabulary name(s) (if provided).

Value

A data frame in long format (one row per annotation).

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
annot <- dictionary.annotations(cqx, taxonomy = "Mlstr_harmo", vocabulary = "status")
opal.logout(o)

## End(Not run)
```

---

harmo.dictionary_apply

Apply the dictionary to a tibble (deprecated)

Description

Deprecated: use dictionary.apply instead.

Usage

```
harmo.dictionary_apply(tibble, variables, categories = NULL)
```

Arguments

tibble	Tibble to be decorated.
variables	A data frame with one row per variable (column name) and then one column per property/attribute.
categories	A data frame with one row per category (columns variable and name) and then column per property/attribute.

Examples

```
## Not run:
data <- tibble::as_tibble(mtcars)
variables <- tibble::tribble(
  ~name, ~valueType, ~`label:en`,  ~`Namespace::Name`, ~unit, ~repeatable, ~index,
  "mpg", "decimal", "Mpg label",  "Value1", "years", 0, 1,
  "cyl", "decimal", "Cyl label",  "Value2", "kg/m2", 0, 2,
  "disp", "decimal", "Disp label", NA, NA, 1, 3
)
categories <- tibble::tribble(
  ~variable, ~name, ~missing, ~`label:en`, ~`label:fr`,
  "cyl", "4", 0, "Four", "Quatre",
  "cyl", "6", 0, "Six", "Six",
  "cyl", "8", 1, "Height", "Huit"
)
data <- dictionary.apply(data, variables, categories)

## End(Not run)
```

`harmo.dictionary_update`

Update the dictionary of a Opal table (deprecated)

Description

Deprecated: use opal.table_dictionary_update instead.

Usage

```
harmo.dictionary_update(opal, project, table, variables, categories = NULL)
```

Arguments

`opal`	Opal connection object.
`project`	Project name where the table will be located.
`table`	Destination table name.
`variables`	A data frame with one row per variable (column name) and then one column per property/attribute (Opal Excel format).
`categories`	A data frame with one row per category (columns variable and name) and then column per property/attribute (Opal Excel format). If there are no categories, this parameter is optional.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
variables <- tibble::tribble(
  ~name, ~valueType, ~`label:en`,  ~`Namespace::Name`, ~unit, ~repeatable, ~index,
  "mpg", "decimal", "Mpg label",  "Value1", "years", 0, 1,
  "cyl", "decimal", "Cyl label",  "Value2", "kg/m2", 0, 2,
  "disp", "decimal", "Disp label", NA, NA, 1, 3
)
categories <- tibble::tribble(
  ~variable, ~name, ~missing, ~`label:en`, ~`label:fr`,
  "cyl", "4", 0, "Four", "Quatre",
  "cyl", "6", 0, "Six", "Six",
  "cyl", "8", 1, "Height", "Huit"
)
opal.table_dictionary_update(o, "test", "mtcars", variables, categories)
opal.logout(o)

## End(Not run)
```

---

harmo.table_get *Get a Opal table as a tibble (deprecated)*

Description

Deprecated: use opal.table_get instead.

Usage

```
harmo.table_get(opal, project, table, variables = NULL, missings = TRUE)
```

Arguments

opal	Opal connection object.
project	Project name where the table is located.
table	Table name from which the tibble should be extracted.
variables	List of variable names or Javascript expression that selects the variables of a table (ignored if value does not refere to a table). See javascript documentation: http://wiki.obiba.org/display/OPALDOC/Variable+Methods
missings	Include the missing values (default is TRUE).

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
opal.logout(o)

## End(Not run)
```

harmo.table_save *Save a local tibble as a Opal table (deprecated)*

Description

Deprecated: use opal.table_save instead.

Usage

```
harmo.table_save(
  opal,
  tibble,
  project,
  table,
  overwrite = TRUE,
  force = FALSE,
  identifiers = NULL,
  policy = "required",
  id.name = "id",
  type = "Participant"
)
```

Arguments

opal	Opal connection object.
tibble	The tibble object to be imported.
project	Project name where the table will be located.
table	Destination table name.
overwrite	If the destination table already exists, it will be replaced (deleted and then imported). Otherwise the table will be updated (data dictionaries merge may conflict). Default is TRUE. See also opal.table_truncate function.
force	If the destination already exists, stop with an informative message if this flag is FALSE (default).
identifiers	Name of the identifiers mapping to use when assigning entities to Opal.
policy	Identifiers policy: 'required' (each identifiers must be mapped prior importation (default)), 'ignore' (ignore unknown identifiers) and 'generate' (generate a system identifier for each unknown identifier).
id.name	The name of the column representing the entity identifiers. Default is 'id'.
type	Entity type (what the data are about). Default is 'Participant'

Value

An invisible logical indicating whether the destination table exists.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
# do some (meta)data transformations, then save in opal's database
opal.table_save(o, cqx, "CPTP", "Cag_coreqx", overwrite = TRUE, force = TRUE)
# or overwrite data only (keep original data dictionary)
opal.table_save(o, cqx, "CPTP", "Cag_coreqx", overwrite = 'values', force = TRUE)
opal.logout(o)

## End(Not run)
```

`oadmin.installed_devtools` *Check devtools package*

Description

Check if devtools package is installed.

Usage

```
oadmin.installed_devtools(opal)
```

Arguments

`opal`	Opal object or list of opal objects.

See Also

Other administration functions: `oadmin.install_bioconductor_package()`, `oadmin.install_devtools()`, `oadmin.install_github_package()`, `oadmin.install_github()`, `oadmin.install_local_package()`, `oadmin.install_package()`, `oadmin.installed_packages()`, `oadmin.installed_package()`, `oadmin.package_description()`, `oadmin.remove_package()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.installed_devtools(o)
opal.logout(o)

## End(Not run)
```

oadmin.installed_package *Check package is installed*

Description

Check package is installed

Usage

```
oadmin.installed_package(opal, pkg)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.

Value

TRUE if installed

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_packages(), oadmin.package_description(), oadmin.remove_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.installed_package(o, 'xxx')
oadmin.installed_package(o, 'stats')
opal.logout(o)

## End(Not run)
```

oadmin.installed_packages

List installed packages

Description

List installed packages

Usage

```
oadmin.installed_packages(opal)
```

Arguments

opal	Opal object or list of opal objects.

Value

The result of the installed.packages() call

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_package(), oadmin.package_description(), oadmin.remove_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.installed_packages(o)
opal.logout(o)

## End(Not run)
```

oadmin.install_bioconductor_package

Install a package from Bioconductor

Description

Install a package from a source repository on GitHub.

Usage

```
oadmin.install_bioconductor_package(opal, pkg)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.

See Also

Other administration functions: oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_packages(), oadmin.installed_package(), oadmin.package_description(), oadmin.remove_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.install_bioconductor_package(o, 'GWASTools')
opal.logout(o)

## End(Not run)
```

---

oadmin.install_devtools

Install devtools package

Description

Install devtools package if not already available.

Usage

```
oadmin.install_devtools(opal)
```

Arguments

opal	Opal object or list of opal objects.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_packages(), oadmin.installed_package(), oadmin.package_description(), oadmin.remove_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.install_devtools(o)
opal.logout(o)

## End(Not run)
```

`oadmin.install_github` *Install a package from GitHub (deprecated)*

Description

Install a package from a source repository on GitHub.

Usage

```
oadmin.install_github(
  opal,
  pkg,
  username = getOption("github.user"),
  ref = "master",
  auth_user = NULL,
  password = NULL
)
```

Arguments

`opal`	Opal object or list of opal objects.
`pkg`	Package name.
`username`	GitHub user or organization name.
`ref`	Desired git reference. Could be a commit, tag, or branch name. Defaults to "master".
`auth_user`	(ignored) Your github username if you're attempting to install a package hosted in a private repository (and your username is different to username).
`password`	(ignored) Your github password

See Also

Other administration functions: `oadmin.install_bioconductor_package()`, `oadmin.install_devtools()`, `oadmin.install_github_package()`, `oadmin.install_local_package()`, `oadmin.install_package()`, `oadmin.installed_devtools()`, `oadmin.installed_packages()`, `oadmin.installed_package()`, `oadmin.package_description()`, `oadmin.remove_package()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.install_github(o, 'opalr', 'obiba')
opal.logout(o)

## End(Not run)
```

`oadmin.install_github_package` *Install a package from GitHub*

Description

Install a package from a source repository on GitHub.

Usage

```
oadmin.install_github_package(
  opal,
  pkg,
  username = getOption("github.user"),
  ref = "master"
)
```

Arguments

`opal`	Opal object or list of opal objects.
`pkg`	Package name.
`username`	GitHub user or organization name.
`ref`	Desired git reference. Could be a commit, tag, or branch name. Defaults to "master".

See Also

Other administration functions: `oadmin.install_bioconductor_package()`, `oadmin.install_devtools()`, `oadmin.install_github()`, `oadmin.install_local_package()`, `oadmin.install_package()`, `oadmin.installed_devtools()`, `oadmin.installed_packages()`, `oadmin.installed_package()`, `oadmin.package_description()`, `oadmin.remove_package()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.install_github_package(o, 'opalr', 'obiba')
opal.logout(o)

## End(Not run)
```

oadmin.install_local_package

Install a package from a local archive file

Description

Install a package from a package archive file. This will upload the archive file and run its installation in the R server.

Usage

```
oadmin.install_local_package(opal, path)
```

Arguments

opal	Opal object or list of opal objects.
path	Path to the package archive file.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_packages(), oadmin.installed_package(), oadmin.package_description(), oadmin.remove_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# install a pre-built local archive file
oadmin.install_local_package(o, '~/Rserve_1.8-7.tar.gz')
# or build archive file from local package source (in current working folder)
oadmin.install_local_package(o, devtools::build())
opal.logout(o)

## End(Not run)
```

oadmin.install_package

Install package

Description

Install package if not already available in Opal(s). To install the latest version of a package, it has to be removed first.

Usage

```
oadmin.install_package(opal, pkg, repos = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
repos	Character vector, the base URLs of the repositories to use.

Value

TRUE if successfully installed

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_local_package(), oadmin.installed_devtools(), oadmin.installed_packages(), oadmin.installed_package(), oadmin.package_description(), oadmin.remove_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.install_package(o, 'xxx')
opal.logout(o)

## End(Not run)
```

---

oadmin.package_description

Get package description

Description

Get package description

Usage

```
oadmin.package_description(opal, pkg, fields = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
fields	A character vector giving the fields to extract from each package's DESCRIPTION file in addition to the default ones, or NULL (default). Unavailable fields result in NA values.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_packages(), oadmin.installed_package(), oadmin.remove_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.package_description(o, 'stats')
opal.logout(o)

## End(Not run)
```

---

oadmin.perm *Get the R permissions*

Description

Get the permissions that were applied to the R service.

Usage

```
oadmin.perm(opal)
```

Arguments

opal	Opal connection object.

Value

A data.frame with columns: subject, type, permission

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.perm(o)
oadmin.perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

oadmin.perm_add *Add or update a R permission*

Description

Add or update a permission on the R service.

Usage

```
oadmin.perm_add(opal, subject, type = "user", permission)
```

Arguments

opal	Opal connection object.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.
permission	The permission to apply: use.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.perm(o)
oadmin.perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

oadmin.perm_delete *Delete a R permission*

Description

Delete a permission that was applied to the R service. Silently returns when there is no such permission.

Usage

```
oadmin.perm_delete(opal, subject, type = "user")
```

Arguments

opal	Opal connection object.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.perm(o)
oadmin.perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

---

oadmin.remove_package *Remove package*

Description

Remove package permanently.

Usage

```
oadmin.remove_package(opal, pkg)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_github(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_packages(), oadmin.installed_package(), oadmin.package_description()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.remove_package(o, 'xxx')
opal.logout(o)

## End(Not run)
```

`opal.annotate` *Apply the annotations to a Opal table*

Description

Set the provided annotations (as the one that can be retrieved from opal.annotations) to the table's data dictionary. Variables that do not exists in the table are ignored.

Usage

```
opal.annotate(opal, datasource, table, annotations)
```

Arguments

`opal`	Opal object.
`datasource`	Name of the datasource.
`table`	Name of the table in the datasource.
`annotations`	A data frame of annotations, with the expected columns: 'variable' (variable name), 'taxonomy' (the taxonomy name), 'vocabulary' (the vocabulary name) and 'term' (the term value, if NULL of NA the annotation is removed).

See Also

Other datasource functions: `opal.annotations()`, `opal.attribute_values()`, `opal.datasources()`, `opal.datasource()`, `opal.tables()`, `opal.table()`, `opal.valueset()`, `opal.variables()`, `opal.variable()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
annots <- opal.annotations(o, 'CPTP', 'Coreqx_final')
opal.annotate(o, 'CPTP', 'Cag_coreqx', annots)
opal.logout(o)

## End(Not run)
```

`opal.annotations` *Get the annotations of a Opal table*

Description

Directly retrieves from the table's data dictionary the variable annotations (attributes with a namespace).

Usage

```
opal.annotations(opal, datasource, table)
```

Arguments

`opal`	Opal object.
`datasource`	Name of the datasource.
`table`	Name of the table in the datasource.

Value

A data frame in long format (one row per annotation).

See Also

Other datasource functions: `opal.annotate()`, `opal.attribute_values()`, `opal.datasources()`, `opal.datasource()`, `opal.tables()`, `opal.table()`, `opal.valueset()`, `opal.variables()`, `opal.variable()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.annotations(o, 'CPTP', 'Coreqx_final')
opal.logout(o)

## End(Not run)
```

opal.assign *Data or expression assignment*

Description

Assign a Opal table, or a R expression or a R object to a R symbol in the current R session.

Usage

```
opal.assign(
  opal,
  symbol,
  value,
  variables = NULL,
  missings = FALSE,
  identifiers = NULL,
  id.name = NULL,
  updated.name = NULL,
  async = FALSE
)
```

Arguments

opal	Opal object or list of opal objects.
symbol	Name of the R symbol.
value	The value to assign evaluated in the following order: a R expression, a function, a fully qualified name of a variable or a table in Opal or any other R object (data.frame, vector).
variables	List of variable names or Javascript expression that selects the variables of a table (ignored if value does not refere to a table). See javascript documentation: http://wiki.obiba.org/display/OPALDOC/Variable+Methods
missings	If TRUE, missing values will be pushed from Opal to R, default is FALSE. Ignored if value is an R expression.
identifiers	Name of the identifiers mapping to use when assigning entities to R (from Opal 2.0).
id.name	Add a vector with the given name representing the entity identifiers (from Opal 2.6). Default is NULL.
updated.name	Add a vector with the given name representing the creation and last update timestamps (from Opal 2.6). Default is NULL.
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other assignment functions: opal.assign.data(), opal.assign.resource(), opal.assign.script(), opal.assign.table.tibble(), opal.assign.table()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# assign a list of variables from table CNSIM1
opal.assign(o, symbol="D", value="datashield.CNSIM1", variables=list("GENDER","LAB_TSC"))
# assign all the variables matching 'LAB' from table HOP of opal object o
opal.assign(o, symbol="D", value="datashield.CNSIM1", variables="name().matches('LAB_')")
# assign a function and call it
opal.assign.script(o, 'hello', quote(function(x) { print(paste0('Hello ', x , '!'))}))
opal.execute(o, "hello('Mr Bean')")
# push an arbitrary data frame to the R server
#opal.assign(o, "D", mtcars)
# push an arbitrary vector to the R server
#opal.assign(o, "C", mtcars$cyl)
opal.logout(o)

## End(Not run)
```

---

opal.assign.data *Data assignment*

Description

Assign a R object to a R symbol in the current R session.

Usage

```
opal.assign.data(opal, symbol, value, async = FALSE)
```

Arguments

opal	Opal object or list of opal objects.
symbol	Name of the R symbol.
value	The R object to assign (data.frame, vector).
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other assignment functions: opal.assign.resource(), opal.assign.script(), opal.assign.table.tibble(), opal.assign.table(), opal.assign()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# push an arbitrary data frame to the R server
opal.assign.data(o, "D", mtcars)
# push an arbitrary vector to the R server
opal.assign.data(o, "C", mtcars$cyl)
# push a string
opal.assign.data(o, "S", "Hello!")
opal.logout(o)

## End(Not run)
```

---

opal.assign.resource *Resource assignment*

Description

Assign a ResourceClient object to a R symbol in the current R session.

Usage

```
opal.assign.resource(opal, symbol, value, async = FALSE)
```

Arguments

opal	Opal object or list of opal objects.
symbol	Name of the R symbol.
value	The fully qualified name of a resource in Opal.
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other assignment functions: opal.assign.data(), opal.assign.script(), opal.assign.table.tibble(), opal.assign.table(), opal.assign()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# assign a resource and make some operation on it
opal.assign.resource(o, "D", "datashield.cram1")
opal.execute(o, "class(D)")
opal.logout(o)

## End(Not run)
```

opal.assign.script *R script assignment*

Description

Assign a R script or expression to a R symbol in the current R session.

Usage

```
opal.assign.script(opal, symbol, value, async = FALSE)
```

Arguments

opal	Opal object or list of opal objects.
symbol	Name of the R symbol.
value	The R expression to assign.
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other assignment functions: opal.assign.data(), opal.assign.resource(), opal.assign.table.tibble(), opal.assign.table(), opal.assign()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# assign a function and call it
opal.assign.script(o, 'hello', quote(function(x) { print(paste0('Hello ', x , '!'))}))
opal.execute(o, "hello('Mr Bean')")
opal.logout(o)

## End(Not run)
```

opal.assign.table *Data assignment to a data.frame*

Description

Assign a Opal table to a data.frame identified by a R symbol in the current R session.

Usage

```
opal.assign.table(
  opal,
  symbol,
  value,
  variables = NULL,
  missings = FALSE,
  identifiers = NULL,
  id.name = NULL,
  updated.name = NULL,
  class = "data.frame",
  async = FALSE
)
```

Arguments

opal	Opal object or list of opal objects.
symbol	Name of the R symbol.
value	The value to assign evaluated in the following order: a fully qualified name of a variable or a table in Opal.
variables	List of variable names or Javascript expression that selects the variables of a table (ignored if value does not refere to a table). See javascript documentation: http://wiki.obiba.org/display/OPALDOC/Variable+Methods
missings	If TRUE, missing values will be pushed from Opal to R, default is FALSE.
identifiers	Name of the identifiers mapping to use when assigning entities to R (from Opal 2.0).
id.name	Add a vector with the given name representing the entity identifiers (from Opal 2.6). Default is NULL.
updated.name	Deprecated. Add a vector with the given name representing the creation and last update timestamps (from Opal 2.6). Default is NULL.
class	The data frame class into which the table is written: can 'data.frame' (default) or 'tibble' (from Opal 2.6 to 2.13) or 'tibble.with.factors' (from Opal 2.14).
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other assignment functions: opal.assign.data(), opal.assign.resource(), opal.assign.script(), opal.assign.table.tibble(), opal.assign()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# assign a list of variables from table CNSIM1
opal.assign.table(o, symbol="D", value="datashield.CNSIM1", variables=list("GENDER","LAB_TSC"))
```

```
opal.execute(o, "colnames(D)")
# assign a table CNSIM1 with a identifiers column
opal.assign.table(o, symbol="H", value="datashield.CNSIM1", id.name="id")
opal.execute(o, "colnames(H)")
# assign all the variables matching 'LAB' from table HOP of opal object o
opal.assign.table(o, symbol="D", value="datashield.CNSIM1", variables="name().matches('LAB_')")
opal.execute(o, "colnames(D)")
opal.logout(o)

## End(Not run)
```

`opal.assign.table.tibble`

Data assignment to a tibble

Description

Assign a Opal table to a tibble identified by a R symbol in the current R session.

Usage

```
opal.assign.table.tibble(
  opal,
  symbol,
  value,
  variables = NULL,
  missings = FALSE,
  identifiers = NULL,
  id.name = "id",
  with.factors = FALSE,
  updated.name = NULL,
  async = FALSE
)
```

Arguments

`opal`	Opal object.
`symbol`	Name of the R symbol.
`value`	The fully qualified name of a table in Opal.
`variables`	List of variable names or Javascript expression that selects the variables of a table (ignored if value does not refere to a table). See javascript documentation: http://wiki.obiba.org/display/OPALDOC/Variable+Methods
`missings`	If TRUE, missing values will be pushed from Opal to R, default is FALSE.
`identifiers`	Name of the identifiers mapping to use when assigning entities to R (from Opal 2.0).

id.name	Add a vector with the given name representing the entity identifiers (from Opal 2.6). Default is 'id'.
with.factors	If TRUE, the categorical variables will be assigned as factors (from Opal 2.14). Default is FALSE.
updated.name	Deprecated. Add a vector with the given name representing the creation and last update timestamps (from Opal 2.6 to 2.13). Default is NULL.
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other assignment functions: opal.assign.data(), opal.assign.resource(), opal.assign.script(), opal.assign.table(), opal.assign()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# assign a table and make some operation on it
opal.assign.table.tibble(o, 'D', 'datashield.CNSIM1')
opal.execute(o, "class(D)")
opal.logout(o)

## End(Not run)
```

---

opal.as_md_table *Array to Markdown*

Description

Helper function for turning an array into its Markdown representation.

Usage

```
opal.as_md_table(
  table,
  icons = TRUE,
  digits = getOption("digits"),
  col.names = colnames(table),
  align,
  caption = NULL
)
```

Arguments

table	An array, including a matrix or a data.frame.
icons	Turn logicals to icons (requires bootstrap style). Default is TRUE.
digits	The maximum number of digits for numeric columns (passed to round()); it can also be a vector of length ncol(table) to set the number of digits for individual columns.
col.names	A character vector of column names to be used in the table
align	The alignment of columns: a character vector consisting of 'l' (left), 'c' (center) and/or 'r' (right); by default, numeric columns are right-aligned, and other columns are left-aligned; if align = NULL, the default alignment is used.
caption	The table caption.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.as_md_table(opal.variables(o, 'datashield', 'CNSIM1'))
opal.logout(o)

## End(Not run)
```

---

opal.attribute_values *Get a vector of values*

Description

Get a vector of values (for each locale) matching the given attribute namespace and name. Vector is null if no such attribute is found.

Usage

```
opal.attribute_values(attributes, namespace = NULL, name = "label")
```

Arguments

attributes	A list of attributes, usually variable or category attributes.
namespace	Optional attribute namespace.
name	Required attribute name.

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.datasources(), opal.datasource(), opal.tables(), opal.table(), opal.valueset(), opal.variables(), opal.variable()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
var <- opal.variable(o, 'datashield', 'CNSIM1', 'GENDER')
opal.attribute_values(var$attributes)
opal.logout(o)

## End(Not run)
```

`opal.command` *Get an asynchronous command*

Description

Get an asynchronous R commands in the remote R session.

Usage

```
opal.command(opal, id, wait = FALSE)
```

Arguments

`opal`	Opal object.
`id`	R command ID.
`wait`	Wait for the command to complete.

See Also

Other command functions: `opal.command_result()`, `opal.command_rm()`, `opal.commands_rm()`, `opal.commands()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.command(o, '1234')
opal.logout(o)

## End(Not run)
```

opal.commands *List the asynchronous commands*

Description

Get the list of asynchronous R commands in the remote R session.

Usage

```
opal.commands(opal, df = TRUE)
```

Arguments

opal	Opal object.
df	Return a data.frame (default is TRUE)

See Also

Other command functions: opal.command_result(), opal.command_rm(), opal.commands_rm(), opal.command()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.commands(o)
opal.logout(o)

## End(Not run)
```

opal.commands_rm *Remove all asynchronous commands*

Description

Remove all asynchronous R commands in the remote R session.

Usage

```
opal.commands_rm(opal)
```

Arguments

opal	Opal object.

See Also

Other command functions: opal.command_result(), opal.command_rm(), opal.commands(), opal.command()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.commands_rm(o)
opal.logout(o)

## End(Not run)
```

---

opal.command_result *Get result of an asynchronous command*

Description

Get the result of an asynchronous R commands in the remote R session. The command is removed from the remote R session after this call.

Usage

```
opal.command_result(opal, id, wait = FALSE)
```

Arguments

opal	Opal object.
id	R command ID.
wait	Wait for the command to complete.

See Also

Other command functions: opal.command_rm(), opal.commands_rm(), opal.commands(), opal.command()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.command_result(o, '1234')
opal.logout(o)

## End(Not run)
```

opal.command_rm *Remove an asynchronous command*

Description

Remove an asynchronous R commands in the remote R session.

Usage

```
opal.command_rm(opal, id)
```

Arguments

opal	Opal object.
id	R command ID.

See Also

Other command functions: opal.command_result(), opal.commands_rm(), opal.commands(), opal.command()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.command_rm(o, '1234')
opal.logout(o)

## End(Not run)
```

opal.datasource *Get a datasource*

Description

Get a datasource

Usage

```
opal.datasource(opal, datasource)
```

Arguments

opal	Opal object.
datasource	Name of the datasource.

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.attribute_values(), opal.datasources(), opal.tables(), opal.table(), opal.valueset(), opal.variables(), opal.variable()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.datasource(o, 'datashield')
opal.logout(o)

## End(Not run)
```

`opal.datasources` *Get datasources*

Description

Get datasources

Usage

```
opal.datasources(opal, df = TRUE)
```

Arguments

opal	Opal object.
df	Return a data.frame (default is TRUE)

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.attribute_values(), opal.datasource(), opal.tables(), opal.table(), opal.valueset(), opal.variables(), opal.variable()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.datasources(o)
opal.logout(o)

## End(Not run)
```

opal.delete *Generic REST resource deletion.*

Description

Generic REST resource deletion.

Usage

```
opal.delete(opal, ..., query = list(), callback = NULL)
```

Arguments

opal	Opal object.
...	Resource path segments.
query	Named list of query parameters.
callback	A callback function to handle the response object.

See Also

Other REST functions: opal.get(), opal.post(), opal.put()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.delete(o, 'some', 'resource')
opal.logout(o)

## End(Not run)
```

opal.execute *Execute a R script*

Description

Execute a R script in the remote R session.

Usage

```
opal.execute(opal, script, async = FALSE)
```

Arguments

opal	Opal object or list of opal objects.
script	R script to execute.
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other execution functions: opal.execute.source(), opal.load_package(), opal.unload_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.execute(o, "x <- 'foo'")
opal.execute(o, "ls()")
opal.logout(o)

## End(Not run)
```

---

opal.execute.source *Execute a R file script*

Description

Upload a R file script and execute it in the remote R session with source().

Usage

```
opal.execute.source(opal, path, async = FALSE)
```

Arguments

opal	Opal object or list of opal objects.
path	Path to the R file script to execute.
async	R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).

See Also

Other execution functions: opal.execute(), opal.load_package(), opal.unload_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.execute.source(o, "myscript.R")
opal.logout(o)

## End(Not run)
```

`opal.file` *Get file content*

Description

Get file content from the Opal file system.

Usage

```
opal.file(opal, path, key = NULL)
```

Arguments

`opal`	Opal object.
`path`	Path to the file in the Opal file system.
`key`	File encryption key: downloaded file will be a zip file with content encrypted (use 7zip to decrypt).

See Also

Other file functions: `opal.file_cp()`, `opal.file_download()`, `opal.file_ls()`, `opal.file_mkdir_tmp()`, `opal.file_mkdir()`, `opal.file_mv()`, `opal.file_read()`, `opal.file_rm()`, `opal.file_upload()`, `opal.file_write()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.file(o, '/home/administrator/joins/join-src-3.csv')
opal.logout(o)

## End(Not run)
```

opal.file_cp *Copy a file*

Description

Copy a file or a folder to another location in the Opal file system.

Usage

```
opal.file_cp(opal, source, destination)
```

Arguments

opal	Opal object.
source	Path to the file in the Opal file system.
destination	New path to the file in the Opal file system.

See Also

Other file functions: opal.file_download(), opal.file_ls(), opal.file_mkdir_tmp(), opal.file_mkdir(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# copy a file to another folder
opal.file_cp(o, '/home/administrator/export/some-data.csv', '/home/userx/deliverables')
# copy recursively a folder to another folder
opal.file_cp(o, '/home/administrator/export', '/home/userx/deliverables')
opal.logout(o)

## End(Not run)
```

opal.file_download *Download a file*

Description

Download a file or a folder from the Opal file system.

Usage

```
opal.file_download(opal, source, destination = NULL, key = NULL)
```

Arguments

opal	Opal object.
source	Path to the file in the Opal file system.
destination	Path to the file to be written. If ommitted, file with same name in the working directory will be written.
key	File encryption key: downloaded file will be a zip file with content encrypted (use 7zip to decrypt).

See Also

Other file functions: opal.file_cp(), opal.file_ls(), opal.file_mkdir_tmp(), opal.file_mkdir(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# download a file
opal.file_download(o, '/home/administrator/joins/join-src-3.csv')
# download a file encrypted by a key: resulting file is a zip with an encrypted content
opal.file_download(o, '/home/administrator/export/some-data.csv',
                   destination='some-data.zip', key='AZF57893FBDE')
# download, create destination folder and rename file
opal.file_download(o, '/home/administrator/spss/DatabaseTest.sav', 'spss/test.sav')
# download a folder
opal.file_download(o, '/home/administrator/export', 'export.zip')
opal.logout(o)

## End(Not run)
```

---

opal.file_ls *List content of a folder*

Description

List content of a folder in the Opal file system.

Usage

```
opal.file_ls(opal, path)
```

Arguments

opal	Opal object.
path	Path to the folder in the Opal file system.

See Also

Other file functions: opal.file_cp(), opal.file_download(), opal.file_mkdir_tmp(), opal.file_mkdir(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# list content of a folder
opal.file_ls(o, '/home/administrator')
opal.logout(o)

## End(Not run)
```

---

opal.file_mkdir *Make a folder*

Description

Make a folder in the Opal file system. Use the parents parameter to ignore if it already exist and to create parent folders.

Usage

```
opal.file_mkdir(opal, path, parents = FALSE)
```

Arguments

opal	Opal object.
path	Path to the new folder in the Opal file system.
parents	No error if existing, make parent directories as needed. Default is FALSE.

See Also

Other file functions: opal.file_cp(), opal.file_download(), opal.file_ls(), opal.file_mkdir_tmp(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# make a folder
opal.file_mkdir(o, '/home/administrator/test', parents = TRUE)
opal.logout(o)

## End(Not run)
```

opal.file_mkdir_tmp *Make a temporary folder*

Description

Make a temporary folder in the Opal file system (make sure it does not exists).

Usage

```
opal.file_mkdir_tmp(opal)
```

Arguments

opal	Opal object.

Value

The path of the created folder.

See Also

Other file functions: opal.file_cp(), opal.file_download(), opal.file_ls(), opal.file_mkdir(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# make a folder
path <- opal.file_mkdir_tmp(o)
opal.logout(o)

## End(Not run)
```

opal.file_mv *Move and/or rename a file*

Description

Move and/or rename a file or a folder in the Opal file system.

Usage

```
opal.file_mv(opal, source, destination)
```

Arguments

opal	Opal object.
source	Path to the file in the Opal file system.
destination	New path to the file in the Opal file system.

See Also

Other file functions: opal.file_cp(), opal.file_download(), opal.file_ls(), opal.file_mkdir_tmp(), opal.file_mkdir(), opal.file_read(), opal.file_rm(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# move a file to another folder
opal.file_mv(o, '/home/administrator/export/some-data.csv', '/home/userx/deliverables')
# rename a file
opal.file_mv(o, '/home/administrator/export/some-data-20170123.csv',
                '/home/administrator/export/some-data.csv')
# move and rename a file
opal.file_mv(o, '/home/administrator/export/some-data-20170123.csv',
                '/home/userx/deliverables/some-data.csv')
opal.logout(o)

## End(Not run)
```

---

opal.file_read *Read a file*

Description

Read a file from the R session workspace into the Opal file system.

Usage

```
opal.file_read(opal, source, destination)
```

Arguments

opal	Opal object.
source	Path to the file in the R session workspace (must exists).
destination	Path to the destination file or folder. Any required sub-folders will be created.

See Also

Other file functions: opal.file_cp(), opal.file_download(), opal.file_ls(), opal.file_mkdir_tmp(), opal.file_mkdir(), opal.file_mv(), opal.file_rm(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# read into folder
opal.file_read(o,"DatabaseTest.sav", "/tmp")
# read and rename
opal.file_read(o,"test/DatabaseTest.sav", "/tmp/Test.sav")
# user home expansion
opal.file_read(o,"DatabaseTest.sav", "~/coucou/pwel.sav")
opal.logout(o)

## End(Not run)
```

---

opal.file_rm *Remove a file*

Description

Remove a file or a folder from the Opal file system.

Usage

```
opal.file_rm(opal, path)
```

Arguments

opal	Opal object.
path	Path to the file in the Opal file system.

See Also

Other file functions: opal.file_cp(), opal.file_download(), opal.file_ls(), opal.file_mkdir_tmp(), opal.file_mkdir(), opal.file_mv(), opal.file_read(), opal.file_upload(), opal.file_write(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# remove a file
opal.file_rm(o, '/home/administrator/export/some-data.csv')
# remove recursively a folder
```

```
opal.file_rm(o, '/home/administrator/export')
opal.logout(o)

## End(Not run)
```

`opal.file_upload` *Upload a file*

Description

Upload a file into the Opal file system.

Usage

```
opal.file_upload(opal, source, destination)
```

Arguments

`opal`	Opal object.
`source`	Path to the file in the local file system.
`destination`	Path of the destination folder in the Opal file system.

See Also

Other file functions: `opal.file_cp()`, `opal.file_download()`, `opal.file_ls()`, `opal.file_mkdir_tmp()`, `opal.file_mkdir()`, `opal.file_mv()`, `opal.file_read()`, `opal.file_rm()`, `opal.file_write()`, `opal.file()`

`opal.file_write` *Write a file*

Description

Write a file from the Opal file system into the R session workspace.

Usage

```
opal.file_write(opal, source, destination = NULL)
```

Arguments

`opal`	Opal object.
`source`	Path to the file in the Opal file system (must exists and be accessible for the user).
`destination`	Path to the destination file, relative to the R session workspace. Any required sub-folders will be created. If ommitted, file with same name will be written.

See Also

Other file functions: opal.file_cp(), opal.file_download(), opal.file_ls(), opal.file_mkdir_tmp(), opal.file_mkdir(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_upload(), opal.file()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# user home expansion
opal.file_write(o, "~/spss/DatabaseTest.sav")
# rename file
opal.file_write(o, "/home/administrator/spss/DatabaseTest.sav", "x.sav")
# create sub-folder
opal.file_write(o, "/home/administrator/spss/DatabaseTest.sav", "test/x.sav")
opal.logout(o)

## End(Not run)
```

---

opal.get *Generic REST resource getter.*

Description

Generic REST resource getter.

Usage

```
opal.get(opal, ..., query = list(), callback = NULL)
```

Arguments

opal	Opal object.
...	Resource path segments.
query	Named list of query parameters.
callback	A callback function to handle the response object.

See Also

Other REST functions: opal.delete(), opal.post(), opal.put()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.get(o, 'project', 'datashield')
opal.logout(o)

## End(Not run)
```

opal.load_package *Load package*

Description

Load package in the remote R session.

Usage

```
opal.load_package(opal, pkg)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.

See Also

Other execution functions: opal.execute.source(), opal.execute(), opal.unload_package()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.load_package(o, 'stats')
opal.logout(o)

## End(Not run)
```

opal.login *Opal login*

Description

Log in Opal(s).

Usage

```
opal.login(
  username = getOption("opal.username"),
  password = getOption("opal.password"),
  token = getOption("opal.token"),
  url = getOption("opal.url"),
  opts = getOption("opal.opts", list()),
  restore = NULL
)
```

Arguments

username	User name in opal(s). Can be provided by "opal.username" option.
password	User password in opal(s). Can be provided by "opal.password" option.
token	Personal access token (since opal 2.15). Only effective if the username or the password is NULL or empty. Can be provided by "opal.token" option.
url	Opal url or list of opal urls. Can be provided by "opal.url" option.
opts	Curl options as described by httr (call httr::httr_options() for details). Can be provided by "opal.opts" option.
restore	Workspace ID to be restored (see also opal.logout)

Value

A opal object or a list of opal objects.

See Also

Other connection functions: `opal.logout()`

Examples

```
## Not run:
#### The below examples illustrate the different ways to login in opal ####

# explicite username/password login
o <- opal.login(username='administrator', password='password', url='https://opal-demo.obiba.org')
opal.logout(o)

 # explicite personal access token login
o <- opal.login(token='HYG16LO0VaX4O0UardNbiqmr2ByBpRke', url='https://opal-demo.obiba.org')
opal.logout(o)

# login using options and user credentials
options(opal.username='administrator',
 opal.password='password',
 opal.url='https://opal-demo.obiba.org')
o <- opal.login()
opal.logout(o)

# login using options and personal access token
options(opal.token='HYG16LO0VaX4O0UardNbiqmr2ByBpRke',
 opal.url='https://opal-demo.obiba.org')
o <- opal.login()
opal.logout(o)

# login using ssl key pair
options(opal.opts=list(
   sslcert='my-publickey.pem',
   sslkey='my-privatekey.pem'))
o <- opal.login(url='https://opal-demo.obiba.org')
opal.logout(o)
```

```
## End(Not run)
```

`opal.logout` *Logout from Opal(s)*

Description

Clear the R sessions and logout from Opal(s).

Usage

```
opal.logout(opal, save = FALSE)
```

Arguments

`opal`	Opal object or a list of opals.
`save`	Save the workspace with given identifier (default value is FALSE, current session ID if TRUE).

See Also

Other connection functions: `opal.login()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.logout(o)

## End(Not run)
```

`opal.post` *Generic REST resource creation.*

Description

Generic REST resource creation.

Usage

```
opal.post(
  opal,
  ...,
  query = list(),
  body = "",
  contentType = "application/x-rscript",
  callback = NULL
)
```

Arguments

opal	Opal object.
...	Resource path segments.
query	Named list of query parameters.
body	The body of the request.
contentType	The type of the body content.
callback	A callback function to handle the response object.

See Also

Other REST functions: opal.delete(), opal.get(), opal.put()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.post(o, 'some', 'resources', body='{"some":"value"}')
opal.logout(o)

## End(Not run)
```

---

opal.project *Get a project*

Description

Get a project

Usage

```
opal.project(opal, project)
```

Arguments

opal	Opal object.
project	Name of the project

See Also

Other project functions: opal.project_create(), opal.project_delete(), opal.project_exists(), opal.projects(), opal.resource_create(), opal.resource_exists(), opal.resource_get(), opal.resources(), opal.resource()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.project(o, 'datashield')
opal.logout(o)

## End(Not run)
```

`opal.projects` *Get projects*

Description

Get projects

Usage

```
opal.projects(opal, df = TRUE)
```

Arguments

`opal`	Opal object.
`df`	Return a data.frame (default is TRUE)

See Also

Other project functions: `opal.project_create()`, `opal.project_delete()`, `opal.project_exists()`, `opal.project()`, `opal.resource_create()`, `opal.resource_exists()`, `opal.resource_get()`, `opal.resources()`, `opal.resource()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.projects(o)
opal.logout(o)

## End(Not run)
```

`opal.project_create` *Create a project*

Description

Create a project

Usage

```
opal.project_create(
  opal,
  project,
  database = NULL,
  title = NULL,
  description = NULL,
  tags = NULL,
  exportFolder = NULL
)
```

Arguments

`opal`	Opal object.
`project`	Name of the project
`database`	The database name (as declared in Opal) to be used to store project's data. If not provided, the project can have views and resources but no raw tables.
`title`	The title of the project (optional).
`description`	The description of the project (optional).
`tags`	A list of tag names (optional).
`exportFolder`	The default location of the exported data files in the Opal file system (optional).

See Also

Other project functions: `opal.project_delete()`, `opal.project_exists()`, `opal.projects()`, `opal.project()`, `opal.resource_create()`, `opal.resource_exists()`, `opal.resource_get()`, `opal.resources()`, `opal.resource()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.project_create(o, 'test', database='opal_data', title='This is a test', tags=list('Test'))
opal.logout(o)

## End(Not run)
```

opal.project_delete *Delete a project*

Description

Delete a project and every data what could have been associated to it.

Usage

```
opal.project_delete(opal, project, archive = FALSE, silent = TRUE)
```

Arguments

opal	Opal object.
project	Name of the project
archive	Logical that is TRUE if the complete removal of the project is requested.
silent	Warn if project does not exist, default is TRUE.

See Also

Other project functions: opal.project_create(), opal.project_exists(), opal.projects(), opal.project(), opal.resource_create(), opal.resource_exists(), opal.resource_get(), opal.resources(), opal.resource()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.project_delete(o, 'test')
opal.logout(o)

## End(Not run)
```

opal.project_exists *Check a project exists*

Description

Check whether a project already exists (and is visible by the requesting user).

Usage

```
opal.project_exists(opal, project)
```

Arguments

opal	Opal object.
project	Name of the project

Value

A logical

See Also

Other project functions: opal.project_create(), opal.project_delete(), opal.projects(), opal.project(), opal.resource_create(), opal.resource_exists(), opal.resource_get(), opal.resources(), opal.resource()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.project_exists(o, 'test')
opal.logout(o)

## End(Not run)
```

---

opal.project_perm *Get the permissions on a project*

Description

Get the permissions that were applied on a project.

Usage

```
opal.project_perm(opal, project)
```

Arguments

opal	Opal connection object.
project	Project name.

Value

A data.frame with columns: subject, type, permission

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.project_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'administrate')
opal.project_perm(o, 'CNSIM')
opal.project_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

`opal.project_perm_add` *Add or update a permission on a project*

Description

Add or update a permission on a project.

Usage

```
opal.project_perm_add(opal, project, subject, type = "user", permission)
```

Arguments

`opal`	Opal connection object.
`project`	Project name where the table will be located.
`subject`	A vector of subject identifiers: user names or group names (depending on the type).
`type`	The type of subject: user (default) or group.
`permission`	The permission to apply: administrate.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.project_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'administrate')
opal.project_perm(o, 'CNSIM')
opal.project_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.project_perm_delete *Delete a permission from a project*

Description

Delete a permission that was applied on a project. Silently returns when there is no such permission.

Usage

```
opal.project_perm_delete(opal, project, subject, type = "user")
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.project_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'administrate')
opal.project_perm(o, 'CNSIM')
opal.project_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.put *Generic REST resource update.*

Description

Generic REST resource update.

Usage

```
opal.put(
  opal,
  ...,
  query = list(),
  body = "",
  contentType = "application/x-rscript",
  callback = NULL
)
```

Arguments

opal	Opal object.
...	Resource path segments.
query	Named list of query parameters.
body	The body of the request.
contentType	The type of the body content.
callback	A callback function to handle the response object.

See Also

Other REST functions: opal.delete(), opal.get(), opal.post()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.put(o, 'some', 'resource', 'toupdate', body='{"some":"value"}')
opal.logout(o)

## End(Not run)
```

---

opal.report *Opal report*

Description

Helper function for generating reports.

Usage

```
opal.report(
  input,
  output = NULL,
  progress = FALSE,
  verbose = FALSE,
  boot_style = NULL
)
```

Arguments

input	Path to the R markdown report file
output	Directory path where to ouput the html report file. Default is the current working directory.
progress	Knitr progress option
verbose	Knitr verbose option
boot_style	Deprecated, directives can be integrated in the YAML header of the R markdown document.

Examples

```
## Not run:
opal.report('input.Rmd', 'report', progress=TRUE)

## End(Not run)
```

`opal.resource` *Get a resource reference of a project*

Description

Get a resource reference of a project

Usage

```
opal.resource(opal, project, resource)
```

Arguments

`opal`	Opal object.
`project`	Name of the project.
`resource`	Name of the resource in the project.

See Also

Other project functions: `opal.project_create()`, `opal.project_delete()`, `opal.project_exists()`, `opal.projects()`, `opal.project()`, `opal.resource_create()`, `opal.resource_exists()`, `opal.resource_get()`, `opal.resources()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource(o, 'RSRC', 'CNSIM1')
opal.logout(o)

## End(Not run)
```

opal.resources *Get the resource references of a project*

Description

Get the resource references of a project

Usage

```
opal.resources(opal, project, df = TRUE)
```

Arguments

opal	Opal object.
project	Name of the project.
df	Return a data.frame (default is TRUE)

See Also

Other project functions: opal.project_create(), opal.project_delete(), opal.project_exists(), opal.projects(), opal.project(), opal.resource_create(), opal.resource_exists(), opal.resource_get(), opal.resource()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resources(o, 'RSRC')
opal.logout(o)

## End(Not run)
```

opal.resources_perm *Get the permissions on any resource*

Description

Get the permissions that were applied globally on the project's resources.

Usage

```
opal.resources_perm(opal, project)
```

Arguments

opal	Opal connection object.
project	The project name.

Value

A data.frame with columns: subject, type, permission

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resources_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'view')
opal.resources_perm(o, 'CNSIM')
opal.resources_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

`opal.resources_perm_add`

Add or update a permission on any resource

Description

Add or update a global permission on the project's resources

Usage

```
opal.resources_perm_add(opal, project, subject, type = "user", permission)
```

Arguments

`opal`	Opal connection object.
`project`	The project name.
`subject`	A vector of subject identifiers: user names or group names (depending on the type).
`type`	The type of subject: user (default) or group.
`permission`	The permission to apply: view, administrate. The 'view' permission is suitable for DataSHIELD operations.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resources_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'view')
opal.resources_perm(o, 'CNSIM')
opal.resources_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.resources_perm_delete *Delete a permission from any resource*

Description

Delete a permission that was applied globally on the project's resources. Silently returns when there is no such permission.

Usage

```
opal.resources_perm_delete(opal, project, subject, type = "user")
```

Arguments

opal	Opal connection object.
project	The project name.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resources_perm_add(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user', 'view')
opal.resources_perm(o, 'CNSIM', 'CNSIM1')
opal.resources_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.resource_create *Create a resource reference in a project*

Description

Create a resource reference in a project

Usage

```
opal.resource_create(
  opal,
  project,
  name,
  url,
  description = NULL,
  format = NULL,
  package = NULL,
  identity = NULL,
  secret = NULL
)
```

Arguments

opal	Opal object.
project	Name of the project.
name	Name of the resource in the project.
url	The URL of the resource.
description	The description of the resource (optional).
format	The format of the data described by the resource (optional).
package	The R package to be loaded prior to the assignment of the resource (optional).
identity	The identity key or username to be used when accessing the resource (optional).
secret	The secret key or password to be used when accessing the resource (optional).

See Also

Other project functions: opal.project_create(), opal.project_delete(), opal.project_exists(), opal.projects(), opal.project(), opal.resource_exists(), opal.resource_get(), opal.resources(), opal.resource()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_create(o, 'RSRC', 'CNSIM4',
  url = 'opal+https://opal-demo.obiba.org/ws/files/projects/RSRC/CNSIM3.zip',
  format = 'csv', secret = 'EeTtQGIob6haio5bx6FUfVvIGkeZJfGq')
opal.logout(o)

## End(Not run)
```

opal.resource_delete *Delete a resource reference*

Description

Removes the reference to a resource. The targeted resource remains untouched.

Usage

```
opal.resource_delete(opal, project, resource, silent = TRUE)
```

Arguments

opal	Opal connection object.
project	Project name where the resource is located.
resource	Resource name to be deleted.
silent	Warn if resource does not exist, default is TRUE.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_delete(o, "RSRC", "CNSIM4")
opal.logout(o)

## End(Not run)
```

opal.resource_exists *Check a resource reference exists*

Description

Check whether a resource already exists in the project (and is visible by the requesting user).

Usage

```
opal.resource_exists(opal, project, resource)
```

Arguments

opal	Opal object.
project	Name of the project.
resource	Name of the resource in the project.

Value

A logical

See Also

Other project functions: opal.project_create(), opal.project_delete(), opal.project_exists(), opal.projects(), opal.project(), opal.resource_create(), opal.resource_get(), opal.resources(), opal.resource()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_exists(o, 'RSRC', 'CNSIM1')
opal.logout(o)

## End(Not run)
```

---

opal.resource_get *Get the resource object of a project*

Description

Get the resource object of a project

Usage

```
opal.resource_get(opal, project, resource)
```

Arguments

opal	Opal object.
project	Name of the project.
resource	Name of the resource in the project.

See Also

Other project functions: opal.project_create(), opal.project_delete(), opal.project_exists(), opal.projects(), opal.project(), opal.resource_create(), opal.resource_exists(), opal.resources(), opal.resource()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
res <- opal.resource_get(o, 'RSRC', 'CNSIM1')
# then interpret locally the resource object (load the appropriate R packages)
library(resourcer)
# coerce to a data.frame
as.data.frame(res)
# or get the resource client object for low-level interactions
rescli <- resourcer::newResourceClient(res)
opal.logout(o)

## End(Not run)
```

`opal.resource_perm` *Get the permissions on a resource*

Description

Get the permissions that were applied on a resource.

Usage

```
opal.resource_perm(opal, project, resource)
```

Arguments

`opal`	Opal connection object.
`project`	The project name.
`resource`	The resource name.

Value

A data.frame with columns: subject, type, permission

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_perm_add(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user', 'view')
opal.resource_perm(o, 'CNSIM', 'CNSIM1')
opal.resource_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.resource_perm_add *Add or update a permission on a resource*

Description

Add or update a permission on a resource

Usage

```
opal.resource_perm_add(
  opal,
  project,
  resource,
  subject,
  type = "user",
  permission
)
```

Arguments

opal	Opal connection object.
project	The project name.
resource	The resource name.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.
permission	The permission to apply: view, administrate. The 'view' permission is suitable for DataSHIELD operations.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_perm_add(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user', 'view')
opal.resource_perm(o, 'CNSIM', 'CNSIM1')
opal.resource_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.resource_perm_delete *Delete a permission from a resource*

Description

Delete a permission that was applied on a resource. Silently returns when there is no such permission.

Usage

```
opal.resource_perm_delete(opal, project, resource, subject, type = "user")
```

Arguments

opal	Opal connection object.
project	The project name.
resource	The resource name.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_perm_add(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user', 'view')
opal.resource_perm(o, 'CNSIM', 'CNSIM1')
opal.resource_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.rm *Remove a R symbol (deprecated)*

Description

Remove a symbol from the current R session. Deprecated: see opal.symbol_rm function instead.

Usage

```
opal.rm(opal, symbol)
```

Arguments

opal	Opal object.
symbol	Name of the R symbol.

See Also

Other symbol functions: opal.symbol_import(), opal.symbol_rm(), opal.symbol_save(), opal.symbols()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.rm(o, 'D')
opal.logout(o)

## End(Not run)
```

---

opal.symbols *List R symbols*

Description

Get the R symbols available in the remote R session.

Usage

```
opal.symbols(opal)
```

Arguments

opal	Opal object.

See Also

Other symbol functions: opal.rm(), opal.symbol_import(), opal.symbol_rm(), opal.symbol_save()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.symbols(o)
opal.logout(o)

## End(Not run)
```

opal.symbol_import *Import a tibble*

Description

Import a tibble identified by the symbol as a table in Opal. This operation creates an importation task in Opal that can be followed (see tasks related functions).

Usage

```
opal.symbol_import(
  opal,
  symbol,
  project,
  identifiers = NULL,
  policy = "required",
  id.name = "id",
  type = "Participant",
  wait = TRUE
)
```

Arguments

opal	Opal object.
symbol	Name of the R symbol representing a tibble.
project	Name of the project into which the data are to be imported.
identifiers	Name of the identifiers mapping to use when assigning entities to Opal.
policy	Identifiers policy: 'required' (each identifiers must be mapped prior importation (default)), 'ignore' (ignore unknown identifiers) and 'generate' (generate a system identifier for each unknown identifier).
id.name	The name of the column representing the entity identifiers. Default is 'id'.
type	Entity type (what the data are about). Default is 'Participant'.
wait	Wait for import task completion. Default is TRUE.

See Also

Other symbol functions: opal.rm(), opal.symbol_rm(), opal.symbol_save(), opal.symbols()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.symbol_import(o, 'D', 'test')
opal.logout(o)

## End(Not run)
```

opal.symbol_rm *Remove a R symbol*

Description

Remove a symbol from the remote R session.

Usage

```
opal.symbol_rm(opal, symbol)
```

Arguments

opal	Opal object.
symbol	Name of the R symbol.

See Also

Other symbol functions: opal.rm(), opal.symbol_import(), opal.symbol_save(), opal.symbols()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.symbol_rm(o, 'D')
opal.logout(o)

## End(Not run)
```

opal.symbol_save *Save a tibble*

Description

Save a tibble identified by symbol as a file of format SAS, SPSS, Stata, CSV or TSV in the remote R session working directory.

Usage

```
opal.symbol_save(opal, symbol, destination)
```

Arguments

opal	Opal object.
symbol	Name of the R symbol representing a tibble.
destination	The path of the file in the R session workspace. Supported file extensions are: .sav (SPSS), .zsav (compressed SPSS), .sas7bdat (SAS), .xpt (SAS Transport), .dta (Stata), .csv (comma separated values), .tsv (tab separated values).

See Also

Other symbol functions: opal.rm(), opal.symbol_import(), opal.symbol_rm(), opal.symbols()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.symbol_save(o, 'D', 'test.sav')
opal.logout(o)

## End(Not run)
```

---

opal.table *Get a table of a datasource*

Description

Get a table of a datasource

Usage

```
opal.table(opal, datasource, table, counts = FALSE)
```

Arguments

opal	Opal object.
datasource	Name of the datasource.
table	Name of the table in the datasource.
counts	Flag to get the number of variables and entities (default is FALSE).

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.attribute_values(), opal.datasources(), opal.datasource(), opal.tables(), opal.valueset(), opal.variables(), opal.variable()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table(o, 'datashield', 'CNSIM1')
opal.logout(o)

## End(Not run)
```

opal.tables *Get tables of a datasource*

Description

Get tables of a datasource

Usage

```
opal.tables(opal, datasource, counts = FALSE, df = TRUE)
```

Arguments

opal	Opal object.
datasource	Name of the datasource.
counts	Flag to get the number of variables and entities (default is FALSE).
df	Return a data.frame (default is TRUE)

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.attribute_values(), opal.datasources(), opal.datasource(), opal.table(), opal.valueset(), opal.variables(), opal.variable()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.tables(o, 'datashield')
opal.logout(o)

## End(Not run)
```

opal.tables_perm *Get the permissions on any table of a project*

Description

Get the permissions that were applied on any table of a project.

Usage

```
opal.tables_perm(opal, project)
```

Arguments

opal	Opal connection object.
project	Project name.

Value

A data.frame with columns: subject, type, permission

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.tables_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'administrate')
opal.tables_perm(o, 'CNSIM')
opal.tables_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

---

opal.tables_perm_add *Add or update a permission on any table of a project*

Description

Add or update a permission on any table of a project.

Usage

```
opal.tables_perm_add(opal, project, subject, type = "user", permission)
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.
permission	The permission to apply: view-values, add, or administrate.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.tables_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'administrate')
opal.tables_perm(o, 'CNSIM')
opal.tables_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.tables_perm_delete *Delete a permission from any table of a project*

Description

Delete a permission that was applied on any table of a project. Silently returns when there is no such permission.

Usage

```
opal.tables_perm_delete(opal, project, subject, type = "user")
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.tables_perm_add(o, 'CNSIM', c('andrei', 'valentina'), 'user', 'administrate')
opal.tables_perm(o, 'CNSIM')
opal.tables_perm_delete(o, 'CNSIM', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.table_create *Create a Opal table or view*

Description

Create a Opal table if it does not already exist. If a list of table references are provided, the table will be a view. The table/view created will have no dictionary, use opal.table_dictionary_update to apply a dictionary.

Usage

```
opal.table_create(opal, project, table, type = "Participant", tables = NULL)
```

Arguments

opal	Opal connection object.
project	Project name where the table is located.
table	Table name to be deleted.
type	Entity type, default is "Participant". Ignored if some table references are provided.
tables	List of the fully qualified table names that are referred by the view.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# make a raw table
opal.table_create(o, "CNSIM", "CNSIM4")
# make a view
opal.table_create(o, "CNSIM", "CNSIM123",
                  tables = c("CNSIM.CNSIM1", "CNSIM.CNSIM2", "CNSIM.CNSIM3"))
opal.logout(o)

## End(Not run)
```

---

opal.table_delete *Delete a Opal table*

Description

Removes both values and data dictionary of a table, or remove the table's logic if the table is a view. Fails if the table does not exist. See also opal.table_truncate.

Usage

```
opal.table_delete(opal, project, table, silent = TRUE)
```

Arguments

opal	Opal connection object.
project	Project name where the table is located.
table	Table name to be deleted.
silent	Warn if table does not exist, default is TRUE.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table_delete(o, "CNSIM", "CNSIM1")
opal.logout(o)

## End(Not run)
```

opal.table_dictionary_get
Get the dictionary of a Opal table

Description

Get the dictionary of a Opal table in a format that can be re-applied with opal.table_dictionary_update.

Usage

```
opal.table_dictionary_get(opal, project, table)
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
table	Table name.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dico <- opal.table_dictionary_get(o, "CNSIM", "CNSIM1")
opal.logout(o)

## End(Not run)
```

opal.table_dictionary_update
Update the dictionary of a Opal table

Description

Directly update the dictionary of a Opal table with the provided dictionary.

Usage

```
opal.table_dictionary_update(
  opal,
  project,
  table,
  variables,
  categories = NULL
)
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
table	Destination table name.
variables	A data frame with one row per variable (column name) and then one column per property/attribute (Opal Excel format).
categories	A data frame with one row per category (columns variable and name) and then column per property/attribute (Opal Excel format). If there are no categories, this parameter is optional.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
variables <- tibble::tribble(
  ~name, ~valueType, ~`label:en`,  ~`Namespace::Name`, ~unit, ~repeatable, ~index,
  "mpg", "decimal", "Mpg label",  "Value1", "years", 0, 1,
  "cyl", "decimal", "Cyl label",  "Value2", "kg/m2", 0, 2,
  "disp", "decimal", "Disp label", NA, NA, 1, 3
)
categories <- tibble::tribble(
  ~variable, ~name, ~missing, ~`label:en`, ~`label:fr`,
  "cyl", "4", 0, "Four", "Quatre",
  "cyl", "6", 0, "Six", "Six",
  "cyl", "8", 1, "Height", "Huit"
)
opal.table_dictionary_update(o, "test", "mtcars", variables, categories)
opal.logout(o)

## End(Not run)
```

---

opal.table_exists *Check a Opal table exists*

Description

Check whether a Opal table exists (and is visible). Optionally check whether the table is a raw table or a view.

Usage

```
opal.table_exists(opal, project, table, view = NA)
```

Arguments

opal	Opal connection object.
project	Project name where the table is located.
table	Table name.
view	Logical to perform an additional check whether the table is a view (TRUE) or a raw table (FALSE). If NULL or NA, the table can be indifferently a view or a raw table. Default is NA.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# check table exists
opal.table_exists(o, "CNSIM", "CNSIM1")
# check table exists AND is a NOT a view
opal.table_exists(o, "CNSIM", "CNSIM1", view = FALSE)
# check table exists AND is a view
opal.table_exists(o, "CNSIM", "CNSIM1", view = TRUE)
opal.logout(o)

## End(Not run)
```

---

opal.table_get *Get a Opal table as a tibble*

Description

Shortcut function to assign a Opal table to a tibble in the R server-side session and then retrieve it into the R client-side session. Requires to have the permission to see the individual values of the table and to perform R assignments.

Usage

```
opal.table_get(opal, project, table, variables = NULL, missings = TRUE)
```

Arguments

opal	Opal connection object.
project	Project name where the table is located.
table	Table name from which the tibble should be extracted.
variables	List of variable names or Javascript expression that selects the variables of a table (ignored if value does not refere to a table). See javascript documentation: http://wiki.obiba.org/display/OPALDOC/Variable+Methods
missings	Include the missing values (default is TRUE).

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
opal.logout(o)

## End(Not run)
```

`opal.table_perm` *Get the permissions on a table*

Description

Get the permissions that were applied on a table.

Usage

```
opal.table_perm(opal, project, table)
```

Arguments

`opal`	Opal connection object.
`project`	Project name where the table will be located.
`table`	Destination table name.

Value

A data.frame with columns: subject, type, permission

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table_perm_add(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user', 'view')
opal.table_perm(o, 'CNSIM', 'CNSIM1')
opal.table_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.table_perm_add *Add or update a permission on a table*

Description

Add or update a permission on a table.

Usage

```
opal.table_perm_add(opal, project, table, subject, type = "user", permission)
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
table	Destination table name.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.
permission	The permission to apply: view, view-values, edit, edit-values, administrate. The 'view' permission is suitable for DataSHIELD operations.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table_perm_add(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user', 'view')
opal.table_perm(o, 'CNSIM', 'CNSIM1')
opal.table_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

opal.table_perm_delete *Delete a permission from a table*

Description

Delete a permission that was applied on a table. Silently returns when there is no such permission.

Usage

```
opal.table_perm_delete(opal, project, table, subject, type = "user")
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
table	Destination table name.
subject	A vector of subject identifiers: user names or group names (depending on the type).
type	The type of subject: user (default) or group.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table_perm_add(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user', 'view')
opal.table_perm(o, 'CNSIM', 'CNSIM1')
opal.table_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

---

opal.table_save *Save a local tibble as a Opal table*

Description

Upload a local tibble to the R server side through Opal, assign this tibble to the provided symbol name and import it as a table into a Opal project.

Usage

```
opal.table_save(
  opal,
  tibble,
  project,
  table,
  overwrite = TRUE,
  force = FALSE,
  identifiers = NULL,
  policy = "required",
  id.name = "id",
  type = "Participant"
)
```

Arguments

opal	Opal connection object.
tibble	The tibble object to be imported.
project	Project name where the table will be located.
table	Destination table name.
overwrite	If the destination table already exists, it will be replaced (deleted, re-created with associated permissions reinstated and then imported). Otherwise the table will be updated (data dictionaries merge may conflict). Default is TRUE. See also opal.table_truncate function.
force	If the destination already exists, stop with an informative message if this flag is FALSE (default).
identifiers	Name of the identifiers mapping to use when assigning entities to Opal.
policy	Identifiers policy: 'required' (each identifiers must be mapped prior importation (default)), ignore' (ignore unknown identifiers) and 'generate' (generate a system identifier for each unknown identifier).
id.name	The name of the column representing the entity identifiers. Default is 'id'.
type	Entity type (what the data are about). Default is 'Participant'

Value

An invisible logical indicating whether the destination table exists.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_get(o, "CPTP", "Cag_coreqx")
# do some (meta)data transformations, then save in opal's database
opal.table_save(o, cqx, "CPTP", "Cag_coreqx", overwrite = TRUE, force = TRUE)
# or overwrite data only (keep original data dictionary)
opal.table_save(o, cqx, "CPTP", "Cag_coreqx", overwrite = 'values', force = TRUE)
opal.logout(o)

## End(Not run)
```

---

opal.table_truncate *Truncate a Opal table*

Description

Removes the values of a table and keep the dictionary untouched. Fails if the table does not exist or is a view. See also opal.table_delete.

Usage

```
opal.table_truncate(opal, project, table)
```

Arguments

opal	Opal connection object.
project	Project name where the table is located.
table	Table name to be truncated.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table_truncate(o, "CNSIM", "CNSIM1")
opal.logout(o)

## End(Not run)
```

---

opal.task *Get a task*

Description

Get the details of a specific task.

Usage

```
opal.task(opal, id)
```

Arguments

opal	Opal object.
id	Task identifier.

See Also

Other task functions: opal.task_cancel(), opal.task_wait(), opal.tasks()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.task(o, '1')
opal.logout(o)

## End(Not run)
```

opal.tasks *Get the tasks*

Description

Get all the tasks with their status at the time of the request.

Usage

```
opal.tasks(opal, df = TRUE)
```

Arguments

opal	Opal object.
df	Return a data.frame (default is TRUE)

See Also

Other task functions: opal.task_cancel(), opal.task_wait(), opal.task()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.tasks(o)
opal.logout(o)

## End(Not run)
```

opal.task_cancel *Cancel a task*

Description

Tries to cancel a task.

Usage

```
opal.task_cancel(opal, id)
```

Arguments

opal	Opal object.
id	Task identifier.

See Also

Other task functions: opal.task_wait(), opal.tasks(), opal.task()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.task_cancel(o, '1')
opal.logout(o)

## End(Not run)
```

---

opal.task_wait *Wait for a task to complete.*

Description

The task completion is defined by its status: *SUCCEEDED*, *FAILED* or *CANCELED*.

Usage

```
opal.task_wait(opal, id, max = NULL)
```

Arguments

opal	Opal object.
id	Task identifier.
max	Maximum time (in seconds) to wait for the task completion. Default is NULL (no maximum).

See Also

Other task functions: opal.task_cancel(), opal.tasks(), opal.task()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.task_wait(o, '1')
opal.logout(o)

## End(Not run)
```

opal.taxonomies *Get taxonomies*

Description

Get all taxonomies. A taxonomy describes the annotations that can be applied to the variables. Taxonomies also drive the variables search interface.

Usage

```
opal.taxonomies(opal, locale = "en", df = TRUE)
```

Arguments

opal	Opal object.
locale	The language for labels (default is "en").
df	Return a data.frame (default is TRUE)

See Also

Other taxonomy functions: opal.taxonomy_delete(), opal.taxonomy_download(), opal.taxonomy_upload(), opal.taxonomy(), opal.terms(), opal.vocabularies(), opal.vocabulary()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.taxonomies(o)
opal.logout(o)

## End(Not run)
```

opal.taxonomy *Get a taxonomy*

Description

Get a specific taxonomy details.

Usage

```
opal.taxonomy(opal, taxonomy)
```

Arguments

opal	Opal object.
taxonomy	Name of the taxonomy.

See Also

Other taxonomy functions: opal.taxonomies(), opal.taxonomy_delete(), opal.taxonomy_download(), opal.taxonomy_upload(), opal.terms(), opal.vocabularies(), opal.vocabulary()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.taxonomy(o, 'Mlstr_area')
opal.logout(o)

## End(Not run)
```

---

opal.taxonomy_delete *Delete a taxonomy*

Description

Delete a taxonomy, without failing if the taxonomy does not exist.

Usage

```
opal.taxonomy_delete(opal, taxonomy)
```

Arguments

opal	Opal object.
taxonomy	Name of the taxonomy.

See Also

Other taxonomy functions: opal.taxonomies(), opal.taxonomy_download(), opal.taxonomy_upload(), opal.taxonomy(), opal.terms(), opal.vocabularies(), opal.vocabulary()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.taxonomy_delete(o, 'Mlstr_area')
opal.logout(o)

## End(Not run)
```

opal.taxonomy_download *Download a taxonomy file*

Description

Download a taxonomy stored in a file in YAML format.

Usage

```
opal.taxonomy_download(opal, taxonomy, destination = NULL)
```

Arguments

opal	Opal object.
taxonomy	Name of the taxonomy.
destination	Path to the taxonomy YAML file. If not provided, the downloaded file will have the taxonomy name with the '.yml' extension and will be located in the working directory.

See Also

Other taxonomy functions: opal.taxonomies(), opal.taxonomy_delete(), opal.taxonomy_upload(), opal.taxonomy(), opal.terms(), opal.vocabularies(), opal.vocabulary()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.taxonomy_download(o, 'Mlstr_area', '~/some/dir/Mlstr_area.yml')
opal.logout(o)

## End(Not run)
```

opal.taxonomy_upload *Upload a taxonomy file*

Description

Upload a taxonomy stored in a local file in YAML format. This operation will fail if the taxonomy already exists, see opal.taxonomy_delete.

Usage

```
opal.taxonomy_upload(opal, path)
```

Arguments

opal	Opal object.
path	Path to the taxonomy YAML file.

See Also

Other taxonomy functions: opal.taxonomies(), opal.taxonomy_delete(), opal.taxonomy_download(), opal.taxonomy(), opal.terms(), opal.vocabularies(), opal.vocabulary()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.taxonomy_upload(o, '~/some/dir/taxo.yml')
opal.logout(o)

## End(Not run)
```

---

opal.terms *Get the terms of a vocabulary*

Description

Get all the terms of a vocabulary. The term describes the value of a variable annotation.

Usage

```
opal.terms(opal, taxonomy, vocabulary, locale = "en", df = TRUE)
```

Arguments

opal	Opal object.
taxonomy	Name of the taxonomy
vocabulary	Name of the vocabulary in the taxonomy
locale	The language for labels (default is "en").
df	Return a data.frame (default is TRUE)

See Also

Other taxonomy functions: opal.taxonomies(), opal.taxonomy_delete(), opal.taxonomy_download(), opal.taxonomy_upload(), opal.taxonomy(), opal.vocabularies(), opal.vocabulary()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.terms(o, 'Mlstr_area', 'Lifestyle_behaviours')
opal.logout(o)

## End(Not run)
```

`opal.unload_package` *Unload package*

Description

Unload package from the remote R session.

Usage

```
opal.unload_package(opal, pkg)
```

Arguments

`opal`	Opal object or list of opal objects.
`pkg`	Package name.

See Also

Other execution functions: `opal.execute.source()`, `opal.execute()`, `opal.load_package()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.unload_package(o, 'stats')
opal.logout(o)

## End(Not run)
```

opal.valueset *Get the values of an entity*

Description

Get the values of an entity in a table.

Usage

```
opal.valueset(opal, datasource, table, identifier)
```

Arguments

opal	Opal object.
datasource	Name of the datasource.
table	Name of the table in the datasource.
identifier	Entity identifier.

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.attribute_values(), opal.datasources(), opal.datasource(), opal.tables(), opal.table(), opal.variables(), opal.variable()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.valueset(o, 'datashield', 'CNSIM1', '1008573362')
opal.logout(o)

## End(Not run)
```

opal.variable *Get a variable of a table*

Description

Get a variable of a table

Usage

```
opal.variable(opal, datasource, table, variable)
```

Arguments

opal	Opal object.
datasource	Name of the datasource.
table	Name of the table in the datasource.
variable	Name of the variable in the table.

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.attribute_values(), opal.datasources(), opal.datasource(), opal.tables(), opal.table(), opal.valueset(), opal.variables()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.variable(o, 'datashield', 'CNSIM1', 'GENDER')
opal.logout(o)

## End(Not run)
```

---

opal.variables *Get variables of a table*

Description

Get variables of a table

Usage

```
opal.variables(opal, datasource, table, locale = "en", df = TRUE)
```

Arguments

opal	Opal object.
datasource	Name of the datasource.
table	Name of the table in the datasource.
locale	The language for labels (default is "en").
df	Return a data.frame (default is TRUE)

See Also

Other datasource functions: opal.annotate(), opal.annotations(), opal.attribute_values(), opal.datasources(), opal.datasource(), opal.tables(), opal.table(), opal.valueset(), opal.variable()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.variables(o, 'datashield', 'CNSIM1')
opal.logout(o)

## End(Not run)
```

`opal.version_compare` *Compare*

Description

Compare Opal version with the provided one. Note that a request must have been done in order to have a non-null Opal version.

Usage

```
opal.version_compare(opal, version)
```

Arguments

`opal`	Opal object.
`version`	The semantic version string to be compared.

Value

>0 if Opal version is more recent, 0 if equals, <0 otherwise.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.version_compare(o, "2.6.0")
opal.logout(o)

## End(Not run)
```

opal.vocabularies *Get the vocabularies of a taxonomy*

Description

Get all the vocabularies of a taxonomy. A vocabulary describes the possible values of variable annotations.

Usage

```
opal.vocabularies(opal, taxonomy, locale = "en", df = TRUE)
```

Arguments

opal	Opal object.
taxonomy	Name of the taxonomy
locale	The language for labels (default is "en").
df	Return a data.frame (default is TRUE)

See Also

Other taxonomy functions: opal.taxonomies(), opal.taxonomy_delete(), opal.taxonomy_download(), opal.taxonomy_upload(), opal.taxonomy(), opal.terms(), opal.vocabulary()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.vocabularies(o, 'Mlstr_area')
opal.logout(o)

## End(Not run)
```

opal.vocabulary *Get a taxonomy vocabulary*

Description

Get a specific vocabulary details.

Usage

```
opal.vocabulary(opal, taxonomy, vocabulary)
```

Arguments

opal	Opal object.
taxonomy	Name of the taxonomy
vocabulary	Name of the vocabulary in the taxonomy

See Also

Other taxonomy functions: opal.taxonomies(), opal.taxonomy_delete(), opal.taxonomy_download(), opal.taxonomy_upload(), opal.taxonomy(), opal.terms(), opal.vocabularies()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.vocabulary(o, 'Mlstr_area', 'Lifestyle_behaviours')
opal.logout(o)

## End(Not run)
```

---

opal.workspaces *Get the R workspaces from a opal.*

Description

Get the R workspaces from a opal.

Usage

```
opal.workspaces(opal)
```

Arguments

opal	Opal object.

See Also

Other workspace functions: opal.workspace_rm(), opal.workspace_save()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.workspaces(o)
opal.logout(o)

## End(Not run)
```

opal.workspace_rm *Remove a R workspace from a opal.*

Description

Remove a R workspace from a opal.

Usage

```
opal.workspace_rm(opal, ws, user = NULL)
```

Arguments

opal	Opal object.
ws	The workspace name
user	The user name associated to the worskpace. If not provided, the current user is applied.

See Also

Other workspace functions: opal.workspace_save(), opal.workspaces()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.workspace_rm(o, 'test')
opal.logout(o)

## End(Not run)
```

opal.workspace_save *Save the current session in a opal R workspace.*

Description

Save the current session in a opal R workspace.

Usage

```
opal.workspace_save(opal, save = TRUE)
```

Arguments

opal	Opal object.
save	Save the workspace with given identifier (default is TRUE, current session ID if TRUE).

See Also

Other workspace functions: opal.workspace_rm(), opal.workspaces()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.workspace_save(o, 'test')
opal.logout(o)

## End(Not run)
```

Index

∗ **DataSHIELD functions**
dsadmin.get_method, 8
dsadmin.get_methods, 9
dsadmin.get_options, 10
dsadmin.install_github_package, 11
dsadmin.install_local_package, 12
dsadmin.install_package, 13
dsadmin.installed_package, 10
dsadmin.package_description, 14
dsadmin.package_descriptions, 15
dsadmin.remove_package, 18
dsadmin.rm_method, 18
dsadmin.rm_methods, 19
dsadmin.rm_option, 20
dsadmin.rm_package_methods, 21
dsadmin.set_method, 21
dsadmin.set_option, 22
dsadmin.set_package_methods, 23
∗ **REST functions**
opal.delete, 56
opal.get, 66
opal.post, 69
opal.put, 76
∗ **administration functions**
oadmin.install_bioconductor_package, 32
oadmin.install_devtools, 33
oadmin.install_github, 34
oadmin.install_github_package, 35
oadmin.install_local_package, 36
oadmin.install_package, 36
oadmin.installed_devtools, 30
oadmin.installed_package, 31
oadmin.installed_packages, 32
oadmin.package_description, 37
oadmin.remove_package, 40
∗ **assignment functions**
opal.assign, 43
opal.assign.data, 44
opal.assign.resource, 45
opal.assign.script, 46
opal.assign.table, 46
opal.assign.table.tibble, 48
∗ **command functions**
opal.command, 51
opal.command_result, 53
opal.command_rm, 54
opal.commands, 52
opal.commands_rm, 52
∗ **connection functions**
opal.login, 67
opal.logout, 69
∗ **datasource functions**
opal.annotate, 41
opal.annotations, 42
opal.attribute_values, 50
opal.datasource, 54
opal.datasources, 55
opal.table, 91
opal.tables, 92
opal.valueset, 111
opal.variable, 111
opal.variables, 112
∗ **execution functions**
opal.execute, 56
opal.execute.source, 57
opal.load_package, 67
opal.unload_package, 110
∗ **file functions**
opal.file, 58
opal.file_cp, 59
opal.file_download, 59
opal.file_ls, 60
opal.file_mkdir, 61
opal.file_mkdir_tmp, 62
opal.file_mv, 62
opal.file_read, 63
opal.file_rm, 64

opal.file_upload, 65
opal.file_write, 65
∗ **project functions**
opal.project, 70
opal.project_create, 72
opal.project_delete, 73
opal.project_exists, 73
opal.projects, 71
opal.resource, 78
opal.resource_create, 81
opal.resource_exists, 83
opal.resource_get, 84
opal.resources, 79
∗ **symbol functions**
opal.rm, 87
opal.symbol_import, 89
opal.symbol_rm, 90
opal.symbol_save, 90
opal.symbols, 88
∗ **task functions**
opal.task, 103
opal.task_cancel, 104
opal.task_wait, 105
opal.tasks, 104
∗ **taxonomy functions**
opal.taxonomies, 106
opal.taxonomy, 106
opal.taxonomy_delete, 107
opal.taxonomy_download, 108
opal.taxonomy_upload, 108
opal.terms, 109
opal.vocabularies, 114
opal.vocabulary, 114
∗ **workspace functions**
opal.workspace_rm, 116
opal.workspace_save, 116
opal.workspaces, 115

dictionary.annotate, 5, *24*
dictionary.annotate.harmo_status, 6, *25*
dictionary.annotations, 6, *25*
dictionary.apply, 7, *26*
dsadmin.get_method, 8, *9–15*, *18–23*
dsadmin.get_methods, *8*, 9, *10–15*, *18–23*
dsadmin.get_options, *8, 9*, 10, *11–15*, *18–23*
dsadmin.install_github_package, *8–11*, 11, *12–15*, *18–23*
dsadmin.install_local_package, *8–12*, 12, *13–15*, *18–23*
dsadmin.install_package, *8–12*, 13, *14, 15*, *18–23*
dsadmin.installed_package, *8–10*, 10, *12–15*, *18–23*
dsadmin.package_description, *8–13*, 14, *15*, *18–23*
dsadmin.package_descriptions, *8–14*, 15, *18–23*
dsadmin.perm, 16
dsadmin.perm_add, 16
dsadmin.perm_delete, 17
dsadmin.remove_package, *8–15*, 18, *19–23*
dsadmin.rm_method, *8–15*, *18*, 18, *19–23*
dsadmin.rm_methods, *8–15*, *18, 19*, 19, *20–23*
dsadmin.rm_option, *8–15*, *18, 19*, 20, *21–23*
dsadmin.rm_package_methods, *8–15*, *18–20*, 21, *22, 23*
dsadmin.set_method, *8–15*, *18–21*, 21, *22, 23*
dsadmin.set_option, *8–15*, *18–22*, 22, *23*
dsadmin.set_package_methods, *8–15*, *18–22*, 23

harmo.annotate, 24
harmo.annotate.status, 25
harmo.annotations, 25
harmo.dictionary_apply, 26
harmo.dictionary_update, 27
harmo.table_get, 28
harmo.table_save, 29

oadmin.install_bioconductor_package, *30–32*, 32, *33–38*, *40*
oadmin.install_devtools, *30–33*, 33, *34–38*, *40*
oadmin.install_github, *30–33*, 34, *35–38*, *40*
oadmin.install_github_package, *30–34*, 35, *36–38*, *40*
oadmin.install_local_package, *30–35*, 36, *37, 38*, *40*
oadmin.install_package, *30–36*, 36, *38*, *40*
oadmin.installed_devtools, 30, *31–38*, *40*
oadmin.installed_package, *30*, 31, *32–38*, *40*

oadmin.installed_packages, *30, 31*, 32, *33–38*, *40*
oadmin.package_description, *30–37*, 37, *40*
oadmin.perm, 38
oadmin.perm_add, 39
oadmin.perm_delete, 39
oadmin.remove_package, *30–38*, 40
opal.annotate, 41, *42*, *50*, *55*, *91, 92*, *111, 112*
opal.annotations, *41*, 42, *50*, *55*, *91, 92*, *111, 112*
opal.as_md_table, 49
opal.assign, 43, *44–47*, *49*
opal.assign.data, *43*, 44, *45–47*, *49*
opal.assign.resource, *43, 44*, 45, *46, 47*, *49*
opal.assign.script, *43–45*, 46, *47*, *49*
opal.assign.table, *43–46*, 46, *49*
opal.assign.table.tibble, *43–47*, 48
opal.attribute_values, *41, 42*, 50, *55*, *91, 92*, *111, 112*
opal.command, 51, *52–54*
opal.command_result, *51–53*, 53, *54*
opal.command_rm, *51–53*, 54
opal.commands, *51*, 52, *53, 54*
opal.commands_rm, *51, 52*, 52, *53, 54*
opal.datasource, *41, 42*, *50*, 54, *55*, *91, 92*, *111, 112*
opal.datasources, *41, 42*, *50*, *55*, 55, *91, 92*, *111, 112*
opal.delete, 56, *66*, *70*, *77*
opal.execute, 56, *57*, *67*, *110*
opal.execute.source, *57*, 57, *67*, *110*
opal.file, 58, *59–66*
opal.file_cp, *58*, 59, *60–66*
opal.file_download, *58, 59*, 59, *61–66*
opal.file_ls, *58–60*, 60, *61–66*
opal.file_mkdir, *58–61*, 61, *62–66*
opal.file_mkdir_tmp, *58–61*, 62, *63–66*
opal.file_mv, *58–62*, 62, *64–66*
opal.file_read, *58–63*, 63, *64–66*
opal.file_rm, *58–64*, 64, *65, 66*
opal.file_upload, *58–64*, 65, *66*
opal.file_write, *58–65*, 65
opal.get, *56*, 66, *70*, *77*
opal.load_package, *57*, 67, *110*
opal.login, 67, *69*
opal.logout, *68*, 69
opal.post, *56*, *66*, 69, *77*
opal.project, 70, *71–74*, *78, 79*, *82*, *84*
opal.project_create, *70, 71*, 72, *73, 74*, *78, 79*, *82*, *84*
opal.project_delete, *70–72*, 73, *74*, *78, 79*, *82*, *84*
opal.project_exists, *70–73*, 73, *78, 79*, *82*, *84*
opal.project_perm, 74
opal.project_perm_add, 75
opal.project_perm_delete, 76
opal.projects, *70*, 71, *72–74*, *78, 79*, *82*, *84*
opal.put, *56*, *66*, *70*, 76
opal.report, 77
opal.resource, *70–74*, 78, *79*, *82*, *84*
opal.resource_create, *70–74*, *78, 79*, 81, *84*
opal.resource_delete, 83
opal.resource_exists, *70–74*, *78, 79*, *82*, 83, *84*
opal.resource_get, *70–74*, *78, 79*, *82*, *84*, 84
opal.resource_perm, 85
opal.resource_perm_add, 86
opal.resource_perm_delete, 87
opal.resources, *70–74*, *78*, 79, *82*, *84*
opal.resources_perm, 79
opal.resources_perm_add, 80
opal.resources_perm_delete, 81
opal.rm, 87, *88–91*
opal.symbol_import, *88*, 89, *90, 91*
opal.symbol_rm, *88, 89*, 90, *91*
opal.symbol_save, *88–90*, 90
opal.symbols, *88*, 88, *89–91*
opal.table, *41, 42*, *50*, *55*, 91, *92*, *111, 112*
opal.table_create, 94
opal.table_delete, 95, *102*
opal.table_dictionary_get, 96
opal.table_dictionary_update, *27*, *94*, 96
opal.table_exists, 97
opal.table_get, *28*, 98
opal.table_perm, 99
opal.table_perm_add, 100
opal.table_perm_delete, 100
opal.table_save, *29*, 101
opal.table_truncate, *29*, *95*, *102*, 102
opal.tables, *41, 42*, *50*, *55*, *91*, 92, *111, 112*
opal.tables_perm, 92
opal.tables_perm_add, 93

opal.tables_perm_delete, 94
opal.task, 103, *104, 105*
opal.task_cancel, *103, 104*, 104, *105*
opal.task_wait, *103–105*, 105
opal.tasks, *103*, 104, *105*
opal.taxonomies, 106, *107–109*, *114, 115*
opal.taxonomy, *106*, 106, *107–109*, *114, 115*
opal.taxonomy_delete, *106, 107*, 107, *108, 109*, *114, 115*
opal.taxonomy_download, *106, 107*, 108, *109*, *114, 115*
opal.taxonomy_upload, *106–108*, 108, *109*, *114, 115*
opal.terms, *106–109*, 109, *114, 115*
opal.unload_package, *57*, *67*, 110
opal.valueset, *41, 42*, *50*, *55*, *91, 92*, 111, *112*
opal.variable, *41, 42*, *50*, *55*, *91, 92*, *111*, 111, *112*
opal.variables, *41, 42*, *50*, *55*, *91, 92*, *111, 112*, 112
opal.version_compare, 113
opal.vocabularies, *106–109*, 114, *115*
opal.vocabulary, *106–109*, *114*, 114
opal.workspace_rm, *115*, 116, *117*
opal.workspace_save, *115, 116*, 116
opal.workspaces, 115, *116, 117*