Package 'opalr'

September 18, 2026

Version 3.7.0

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

Depends R (>= 3.1), httr

Imports jsonlite, readr, mime, progress, labelled, tibble

Suggests e1071, knitr, knitrBootstrap, rmarkdown, 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://github.com/obiba/opalr/, https://www.obiba.org/opalr/, https://www.obiba.org/pages/products/opal/, https://academic.oup.com/ije/article/46/5/1372/4102813, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008880, https://datashield.org/

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

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-09-18 18:10:02 UTC

Contents

dictionary.annotate . . . 6
dictionary.annotate.harmo_status . . . 7
dictionary.annotations . . . 8
dictionary.apply . . . 9
dictionary.inspect . . . 10
dsadmin.activity . . . 11
dsadmin.activity_summary . . . 12
dsadmin.get_method . . . 13
dsadmin.get_methods . . . 14
dsadmin.get_options . . . 14
dsadmin.installed_package . . . 15
dsadmin.install_github_package . . . 16
dsadmin.install_local_package . . . 17
dsadmin.install_package . . . 18
dsadmin.log . . . 19
dsadmin.package_description . . . 21
dsadmin.package_descriptions . . . 22
dsadmin.perm . . . 23
dsadmin.perm_add . . . 23
dsadmin.perm_delete . . . 24
dsadmin.profile . . . 25
dsadmin.profiles . . . 25
dsadmin.profile_access . . . 26
dsadmin.profile_create . . . 27
dsadmin.profile_delete . . . 28
dsadmin.profile_enable . . . 29
dsadmin.profile_exists . . . 29
dsadmin.profile_init . . . 30
dsadmin.profile_perm . . . 31
dsadmin.profile_perm_add . . . 32
dsadmin.profile_perm_delete . . . 33
dsadmin.profile_rparser . . . 34
dsadmin.publish_package . . . 35
dsadmin.quota . . . 36
dsadmin.quotas . . . 37
dsadmin.quota_delete . . . 38
dsadmin.quota_enable . . . 39
dsadmin.quota_exists . . . 40
dsadmin.quota_save . . . 41
dsadmin.quota_usage . . . 42
dsadmin.remove_package . . . 43

dsadmin.rm_method . . . 44
dsadmin.rm_methods . . . 45
dsadmin.rm_option . . . 45
dsadmin.rm_options . . . 46
dsadmin.rm_package_methods . . . 47
dsadmin.set_method . . . 48
dsadmin.set_option . . . 49
dsadmin.set_package_methods . . . 50
dsadmin.unpublish_package . . . 51
oadmin.activity . . . 52
oadmin.activity_summary . . . 53
oadmin.installed_devtools . . . 54
oadmin.installed_package . . . 54
oadmin.installed_packages . . . 55
oadmin.install_bioconductor_package . . . 56
oadmin.install_cran_package . . . 57
oadmin.install_devtools . . . 57
oadmin.install_github_package . . . 58
oadmin.install_local_package . . . 59
oadmin.install_package . . . 60
oadmin.log . . . 61
oadmin.log_rest . . . 61
oadmin.log_sql . . . 62
oadmin.package_description . . . 63
oadmin.perm . . . 64
oadmin.perm_add . . . 64
oadmin.perm_delete . . . 65
oadmin.remove_package . . . 66
oadmin.r_perm . . . 66
oadmin.r_perm_add . . . 67
oadmin.r_perm_delete . . . 68
oadmin.system_metrics . . . 68
oadmin.system_perm . . . 69
oadmin.system_perm_add . . . 70
oadmin.system_perm_delete . . . 70
oadmin.users . . . 71
oadmin.user_add . . . 72
oadmin.user_delete . . . 72
oadmin.user_enable . . . 73
oadmin.user_exists . . . 74
oadmin.user_profiles . . . 75
oadmin.user_profile_delete . . . 75
oadmin.user_reset_password . . . 76
opal.annotate . . . 77
opal.annotations . . . 78
opal.assign . . . 79
opal.assign.data . . . 80
opal.assign.resource . . . 81

opal.assign.script . . . 82
opal.assign.table . . . 82
opal.assign.table.tibble . . . 84
opal.as_md_table . . . 85
opal.attribute_values . . . 86
opal.command . . . 87
opal.commands . . . 88
opal.commands_rm . . . 88
opal.command_result . . . 89
opal.command_rm . . . 90
opal.datasource . . . 90
opal.datasources . . . 91
opal.delete . . . 92
opal.execute . . . 92
opal.execute.source . . . 93
opal.file . . . 94
opal.file_cp . . . 95
opal.file_download . . . 95
opal.file_info . . . 96
opal.file_ls . . . 97
opal.file_mkdir . . . 98
opal.file_mkdir_tmp . . . 99
opal.file_mv . . . 99
opal.file_read . . . 100
opal.file_rm . . . 101
opal.file_unzip . . . 102
opal.file_upload . . . 103
opal.file_write . . . 104
opal.get . . . 105
opal.load_package . . . 106
opal.login . . . 106
opal.logout . . . 108
opal.perms . . . 109
opal.post . . . 110
opal.profiles . . . 111
opal.project . . . 112
opal.projects . . . 112
opal.projects_databases . . . 113
opal.project_backup . . . 114
opal.project_command . . . 115
opal.project_create . . . 115
opal.project_delete . . . 117
opal.project_exists . . . 117
opal.project_perm . . . 118
opal.project_perm_add . . . 119
opal.project_perm_delete . . . 120
opal.project_restore . . . 120
opal.put . . . 121

opal.quotas . . . 122
opal.report . . . 123
opal.resource . . . 124
opal.resources . . . 125
opal.resources_perm . . . 125
opal.resources_perm_add . . . 126
opal.resources_perm_delete . . . 127
opal.resource_create . . . 128
opal.resource_delete . . . 129
opal.resource_exists . . . 130
opal.resource_extension_create . . . 131
opal.resource_get . . . 132
opal.resource_perm . . . 133
opal.resource_perm_add . . . 134
opal.resource_perm_delete . . . 135
opal.resource_view_create . . . 136
opal.resource_view_reconnect . . . 137
opal.rm . . . 138
opal.session . . . 138
opal.session_delete . . . 139
opal.session_events . . . 140
opal.session_exists . . . 140
opal.session_get . . . 141
opal.session_running . . . 142
opal.sql . . . 143
opal.sql_history . . . 144
opal.symbols . . . 145
opal.symbol_import . . . 145
opal.symbol_rm . . . 146
opal.symbol_save . . . 147
opal.table . . . 148
opal.tables . . . 149
opal.tables_perm . . . 149
opal.tables_perm_add . . . 150
opal.tables_perm_delete . . . 151
opal.table_create . . . 151
opal.table_delete . . . 152
opal.table_dictionary_get . . . 153
opal.table_dictionary_update . . . 154
opal.table_exists . . . 155
opal.table_export . . . 156
opal.table_get . . . 157
opal.table_import . . . 158
opal.table_perm . . . 159
opal.table_perm_add . . . 160
opal.table_perm_delete . . . 161
opal.table_save . . . 162
opal.table_truncate . . . 163

opal.table_view_create . . . 164
opal.table_view_update . . . 165
opal.task . . . 166
opal.tasks . . . 167
opal.task_cancel . . . 167
opal.task_delete . . . 168
opal.task_wait . . . 169
opal.taxonomies . . . 169
opal.taxonomy . . . 170
opal.taxonomy_delete . . . 171
opal.taxonomy_download . . . 171
opal.taxonomy_upload . . . 172
opal.terms . . . 173
opal.token . . . 174
opal.tokens . . . 174
opal.token_datashield_create . . . 175
opal.token_delete . . . 176
opal.token_renew . . . 177
opal.token_r_create . . . 177
opal.token_sql_create . . . 178
opal.unload_package . . . 179
opal.valueset . . . 180
opal.variable . . . 181
opal.variables . . . 181
opal.variable_summary . . . 182
opal.version_compare . . . 183
opal.vocabularies . . . 184
opal.vocabulary . . . 185
opal.workspaces . . . 185
opal.workspace_restore . . . 186
opal.workspace_rm . . . 187
opal.workspace_save . . . 187

Index **189**

`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, merge = FALSE)
```

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.
merge	Either append attributes to existing ones or replace them. Default is FALSE, for dictionary consistency.

Examples

```
## Not run:
data <- tibble::as_tibble(mtcars)
variables <- tibble::tribble(
 ~name, ~valueType, ~`label:en`, ~`label:fr`,  ~`Namespace::Name`, ~unit, ~repeatable, ~index,
  "mpg", "decimal", "Mpg label", "Mpg libellé",  "Value1", "years", 0, 1,
  "cyl", "decimal", "Cyl label", "Cyl libellé",  "Value2", "kg/m2", 0, 2,
  "disp", "decimal", "Disp label", "Disp libellé", 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)
```

`dictionary.inspect` *Inspect the dictionary of a tibble*

Description

Inspect the data dictionary, checking for inconsistencies. Error is raised when the identifiers column cannot be found.

Usage

```
dictionary.inspect(tibble, id.name = "id", warn = TRUE)
```

Arguments

`tibble`	Tibble to be inspected.
`id.name`	The name of the column representing the entity identifiers. Default is 'id'.
`warn`	Print warning messages. Default is TRUE.

Value

TRUE if inspection is successful, FALSE otherwise.

Examples

```
## Not run:
# prepare datasets of visits, several visits for one patient
visits <- tibble::tribble(
~id, ~patient_id, ~sex, ~visit_date,
1, 1, "M", as.Date("2020-01-01"),
```

```
2, 2, "F", as.Date("2020-01-02"),
3, 3, "M", as.Date("2020-01-03"),
4, 3, "M", as.Date("2020-01-04"))

o <- opal.login("administrator", "password", url = "https://opal-demo.obiba.org")

# save visits dataset
opal.table_save(o, visits, "RSRC", "visits", type = "Visit", force = TRUE)

# get visit and make it a dataset of patients
patients <- opal.table_get(o, "RSRC", "visits", id.name = "id")
# set dataset IDs
patients$visit_id <- patients$id
patients$id <- patients$patient_id
patients$patient_id <- NULL
patients

# save patients dataset, there should be a warning that some variables are not repeatable
# while there are patients with multiple data lines
opal.table_save(o, patients, "RSRC", "patients", type = "Participant", force = TRUE)

opal.logout(o)

## End(Not run)
```

`dsadmin.activity` *Get DataSHIELD activity*

Description

Get the recorded DataSHIELD session metrics.

Usage

```
dsadmin.activity(
  opal,
  user = NULL,
  profile = NULL,
  from = NULL,
  to = NULL,
  df = TRUE
)
```

Arguments

`opal`	Opal connection object.
`user`	Optional user name.
`profile`	Optional profile name.

from	Optional start date.
to	Optional end date.
df	Return a data.frame (default is TRUE)

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# all sessions metrics
dsadmin.activity(o)
# user and profile specific session metrics
dsadmin.activity(o, user = 'dsuser', profile = 'default')
# user sessions in a time range
dsadmin.activity(o, user = "dsuser", from = "2022-07-01", to = "2023-01-01")
opal.logout(o)

## End(Not run)
```

---

dsadmin.activity_summary

Get DataSHIELD activity summary

Description

Get the recorded DataSHIELD session metrics, grouped by profile and user.

Usage

```
dsadmin.activity_summary(
  opal,
  user = NULL,
  profile = NULL,
  from = NULL,
  to = NULL,
  df = TRUE
)
```

Arguments

opal	Opal connection object.
user	Optional user name.
profile	Optional profile name.
from	Optional start date.
to	Optional end date.
df	Return a data.frame (default is TRUE)

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# all sessions metrics
dsadmin.activity_summary(o)
# user and profile specific session metrics
dsadmin.activity_summary(o, user = 'dsuser', profile = 'default')
# user sessions in a time range
dsadmin.activity_summary(o, user = "dsuser", from = "2022-07-01", to = "2023-01-01")
opal.logout(o)

## End(Not run)
```

---

dsadmin.get_method *Get a DataSHIELD method*

Description

Get a DataSHIELD method

Usage

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

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"
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

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_description(), dsadmin.package_descriptions(), dsadmin.publish_package(),
dsadmin.remove_package(), dsadmin.rm_method(), dsadmin.rm_methods(), dsadmin.rm_option(),
dsadmin.rm_options(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(),
dsadmin.set_package_methods(), dsadmin.unpublish_package()

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", profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
type	Type of the method: "aggregate" (default) or "assign"
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

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_description(), dsadmin.package_descriptions(), dsadmin.publish_package(), dsadmin.remove_package(), dsadmin.rm_method(), dsadmin.rm_methods(), dsadmin.rm_option(), dsadmin.rm_options(), dsadmin.rm_package_methods(), dsadmin.set_method(), dsadmin.set_option(), dsadmin.set_package_methods(), dsadmin.unpublish_package()

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, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

Value

TRUE if installed

See Also

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

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",
  profile = NULL
)
```

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'.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

Value

TRUE if installed

See Also

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

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, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
path	Path to the package archive, ending with .
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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,
  profile = 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.
`profile`	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

Value

TRUE if installed

See Also

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

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.log *Get DataSHIELD logs*

Description

The DataSHIELD log is structured as follows:

Usage

```
dsadmin.log(opal, all = TRUE)
```

Arguments

opal	Opal connection object.
all	Get all or only latest log messages.

Details

Common fields

- timestamp: when action is logged
- version: not used
- message: human readable message
- logger_name: name of the log channel
- thread_name: java thread name
- level: log level (TRACE, DEBUG, INFO, WARN, ERROR)
- level_value: log level numeric value
- ip: ip origin of the request, when available

- ds_id: unique DS session ID
- ds_profile: DS profile name (available after the R server session is created, can be after authentication)
- username: DS user name
- r_duration: time spent by the remote R server action (all except PARSE)
- r_size: R result serialized object size in bytes
- ds_action: DS operation (OPEN, CLOSE, RESOLVE, PARSE, ASSIGN, AGGREGATE, LS, RM, WS_SAVE, WS_RESTORE)

DS operations fields

- OPEN: DS session opened (note: happens after successful authentication as R server session creation is deferred)
- CLOSE: DS session closed
- RESOLVE: Table or resource access is resolved
  - ds_symbol: symbol name to assign
  - ds_table: table name to assign
  - ds_resource: resource name to assign
- PARSE: R expression parsed
  - ds_map: DS function mappings used in the R parser, separated by semicolons when several functions are called
  - ds_script_in: R script as sent by DS user
  - ds_script_out: R script rewritten by opal, to be evaluated
- ASSIGN:
  - ds_symbol: assigned symbol name
  - ds_table: table name that is assigned
  - ds_resource: resource name that is assigned
  - ds_eval: assign R expression that is evaluated
- AGGREGATE:
  - ds_eval: aggregate R expression that is evaluated
- LS: R symbols listed
- RM: R symbol removed
  - ds_symbol: symbol name to remove
- WS_SAVE:
  - ds_ws: workspace name
- WS_RESTORE:
  - ds_ws: workspace name

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.log(o)
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, profile = 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.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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, profile = NULL)
```

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)
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

Value

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

See Also

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

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.profile *Get a DataSHIELD profile*

Description

Note that getting a specific DataSHIELD profile details is not allowed for regular DataSHIELD users when the profile has no restricted access. This function is for profiles editors only (system administrators or DataSHIELD administrators).

Usage

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

Arguments

opal	Opal object.
name	Name of the profile.

See Also

Other DataSHIELD profiles: dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

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

## End(Not run)
```

dsadmin.profiles *Get DataSHIELD profiles*

Description

Get DataSHIELD profiles

Usage

```
dsadmin.profiles(opal, df = TRUE)
```

Arguments

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

Value

The DataSHIELD profiles as a data.frame or a list

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser()

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.profile_access

Restrict or open access to a DataSHIELD profile

Description

When access is restricted, only users (or group of users) with this profile use permissions will be allowed to use this profile. When access is not restricted, all DataSHIELD users are allowed to use this profile. See also dsadmin.profile_perm.

Usage

```
dsadmin.profile_access(opal, name, restricted = TRUE)
```

Arguments

opal	Opal object.
name	Name of the profile.
restricted	Default value is TRUE.

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.profile_create(o, name = 'survival', cluster = 'demo')
dsadmin.profile_access(o, name = 'survival', restricted = TRUE)
opal.logout(o)

## End(Not run)
```

---

dsadmin.profile_create

Create a DataSHIELD profile

Description

The created DataSHIELD profile will not be enabled and no access restrictions is applied.

Usage

```
dsadmin.profile_create(opal, name, cluster = "default", rParser = NULL)
```

Arguments

opal	Opal object.
name	Name of the profile.
cluster	Name of the R servers cluster to which the profile will be attached to. Default value is 'default'.
rParser	Version of the DataSHIELD R parser that applies to this profile. If not specified, the system's default one will be used. A valid version would be one of 'v1' or 'v2'.

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

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

## End(Not run)
```

`dsadmin.profile_delete` *Delete a DataSHIELD profile*

Description

Delete a DataSHIELD profile

Usage

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

Arguments

`opal`	Opal object.
`name`	Name of the profile.

See Also

Other DataSHIELD profiles: `dsadmin.profile()`, `dsadmin.profile_access()`, `dsadmin.profile_create()`, `dsadmin.profile_enable()`, `dsadmin.profile_exists()`, `dsadmin.profile_init()`, `dsadmin.profile_perm()`, `dsadmin.profile_perm_add()`, `dsadmin.profile_perm_delete()`, `dsadmin.profile_rparser()`, `dsadmin.profiles()`

Examples

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

## End(Not run)
```

dsadmin.profile_enable *Enable or disable a DataSHIELD profile*

Description

Enable or disable a DataSHIELD profile

Usage

```
dsadmin.profile_enable(opal, name, enabled = TRUE)
```

Arguments

opal	Opal object.
name	Name of the profile.
enabled	Default value is TRUE.

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.profile_create(o, name = 'survival', cluster = 'demo')
dsadmin.profile_enable(o, name = 'survival', enabled = TRUE)
opal.logout(o)

## End(Not run)
```

dsadmin.profile_exists *Check a DataSHIELD profile exists*

Description

Check a DataSHIELD profile exists

Usage

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

Arguments

opal	Opal object.
name	Name of the profile.

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
if (!dsadmin.profile_exists(o, name = 'survival'))
  dsadmin.profile_create(o, name = 'survival', cluster = 'demo')
opal.logout(o)

## End(Not run)
```

---

dsadmin.profile_init *Initialize a DataSHIELD profile*

Description

Clean the DataSHIELD's profile settings from all methods and options (including custom ones). These settings are then repopulated with installed DataSHIELD R packages settings, optionaly filtered by the name. See also dsadmin.publish_package, dsadmin.set_package_methods or dsadmin.set_option.

Usage

```
dsadmin.profile_init(opal, name, packages = NULL)
```

Arguments

opal	Opal object.
name	Name of the profile.
packages	A list DataSHIELD R package names

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.profile_create(o, name = 'survival', cluster = 'demo')
dsadmin.profile_init(o, name = 'survival', packages = c('dsSurvival'))
opal.logout(o)

## End(Not run)
```

---

dsadmin.profile_perm *Get the permissions of a DataSHIELD profile*

Description

Get the permissions of a DataSHIELD profile

Usage

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

Arguments

opal	Opal connection object.
name	Profile name.

Value

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

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

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

## End(Not run)
```

`dsadmin.profile_perm_add` *Add or update a permission on a DataSHIELD profile*

Description

When adding/updating some permissions on a DataSHIELD profile, this profile is automatically set in restricted access mode.

Usage

```
dsadmin.profile_perm_add(
  opal,
  name,
  subject,
  type = "user",
  permission = "use"
)
```

Arguments

`opal`	Opal connection object.
`name`	Profile.
`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.

See Also

Other DataSHIELD profiles: `dsadmin.profile()`, `dsadmin.profile_access()`, `dsadmin.profile_create()`, `dsadmin.profile_delete()`, `dsadmin.profile_enable()`, `dsadmin.profile_exists()`, `dsadmin.profile_init()`, `dsadmin.profile_perm()`, `dsadmin.profile_perm_delete()`, `dsadmin.profile_rparser()`, `dsadmin.profiles()`

Examples

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

## End(Not run)
```

dsadmin.profile_perm_delete *Delete a permission from a DataSHIELD profile*

Description

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

Usage

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

Arguments

opal	Opal connection object.
name	Profile 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.

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_rparser(), dsadmin.profiles()

Examples

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

## End(Not run)
```

dsadmin.profile_rparser *Set or remove the R parser version of a DataSHIELD profile*

Description

Set or remove the R parser version of a DataSHIELD profile

Usage

```
dsadmin.profile_rparser(opal, name, rParser = NULL)
```

Arguments

opal	Opal object.
name	Name of the profile.
rParser	Version of the DataSHIELD R parser that applies to this profile. If not specified, the system's default one will be used. A valid version would be one of 'v1' or 'v2'.

See Also

Other DataSHIELD profiles: dsadmin.profile(), dsadmin.profile_access(), dsadmin.profile_create(), dsadmin.profile_delete(), dsadmin.profile_enable(), dsadmin.profile_exists(), dsadmin.profile_init(), dsadmin.profile_perm(), dsadmin.profile_perm_add(), dsadmin.profile_perm_delete(), dsadmin.profiles()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.profile_create(o, name = 'survival', cluster = 'demo')
# apply R parser version v2
dsadmin.profile_rparser(o, name = 'survival', rParser = 'v2')
# apply system's default R parser version
dsadmin.profile_rparser(o, name = 'survival')
opal.logout(o)

## End(Not run)
```

dsadmin.publish_package *Publish DataSHIELD package settings*

Description

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

Usage

```
dsadmin.publish_package(opal, pkg, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

Value

TRUE if successfull

See Also

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

Examples

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

## End(Not run)
```

dsadmin.quota *Get a DataSHIELD usage quota*

Description

Get the DataSHIELD usage quota that was defined for a subject and a usage metric. Note that this is the quota as it was declared and not necessarily the one that applies to a user: see dsadmin.quota_usage for the resolved one.

Usage

```
dsadmin.quota(opal, metric = "EXECUTION_TIME", type = "system", subject = NULL)
```

Arguments

opal	Opal object.
metric	The usage metric: 'EXECUTION_TIME' (default) is the R server's cpu time consumed, 'SESSION_TIME' is the time R sessions are held, whether they compute or not.
type	The type of subject: system (default), user or group.
subject	The user name or the group name, not applicable when the type is system.

Details

Requires DataSHIELD or system administration permissions.

Value

The DataSHIELD quota as a list, or NULL when there is no such quota.

See Also

Other DataSHIELD quotas: dsadmin.quota_delete(), dsadmin.quota_enable(), dsadmin.quota_exists(), dsadmin.quota_save(), dsadmin.quota_usage(), dsadmin.quotas(), opal.quotas()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.quota(o, metric = 'EXECUTION_TIME', type = 'user', subject = 'andrei')
opal.logout(o)

## End(Not run)
```

`dsadmin.quotas` *Get the DataSHIELD usage quotas*

Description

Get the R usage quotas that apply to the DataSHIELD execution context. A quota is an allowance of R usage, given to a subject (the system default, a group or a user) for a usage metric. Having no quota means unlimited usage, which is not the same as a quota with a limit of zero (that one forbids DataSHIELD altogether for the subject).

Usage

```
dsadmin.quotas(opal, df = TRUE)
```

Arguments

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

Details

Requires DataSHIELD or system administration permissions.

Value

The DataSHIELD quotas as a data.frame (columns are: id, context, type, subject, metric, period, limit, enabled, where limit is expressed in minutes) or a list.

See Also

Other DataSHIELD quotas: `dsadmin.quota()`, `dsadmin.quota_delete()`, `dsadmin.quota_enable()`, `dsadmin.quota_exists()`, `dsadmin.quota_save()`, `dsadmin.quota_usage()`, `opal.quotas()`

Examples

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

## End(Not run)
```

`dsadmin.quota_delete` *Delete a DataSHIELD usage quota*

Description

Delete the DataSHIELD usage quota of a subject for a usage metric. Silently returns when there is no such quota. Deleting the quota that applied makes the subject unlimited again, unless a broader quota still matches.

Usage

```
dsadmin.quota_delete(
  opal,
  metric = "EXECUTION_TIME",
  type = "system",
  subject = NULL
)
```

Arguments

`opal`	Opal object.
`metric`	The usage metric: 'EXECUTION_TIME' (default) or 'SESSION_TIME'.
`type`	The type of subject: system (default), user or group.
`subject`	The user name or the group name, not applicable when the type is system.

Details

Requires DataSHIELD or system administration permissions.

See Also

Other DataSHIELD quotas: `dsadmin.quota()`, `dsadmin.quota_enable()`, `dsadmin.quota_exists()`, `dsadmin.quota_save()`, `dsadmin.quota_usage()`, `dsadmin.quotas()`, `opal.quotas()`

Examples

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

## End(Not run)
```

`dsadmin.quota_enable` *Enable or disable a DataSHIELD usage quota*

Description

A disabled quota is kept but does not apply: the subject falls back on the quota of its groups or on the system default, and is unlimited when there is none.

Usage

```
dsadmin.quota_enable(
  opal,
  enabled = TRUE,
  metric = "EXECUTION_TIME",
  type = "system",
  subject = NULL
)
```

Arguments

`opal`	Opal object.
`enabled`	Default value is TRUE.
`metric`	The usage metric: 'EXECUTION_TIME' (default) or 'SESSION_TIME'.
`type`	The type of subject: system (default), user or group.
`subject`	The user name or the group name, not applicable when the type is system.

Details

Requires DataSHIELD or system administration permissions.

See Also

Other DataSHIELD quotas: `dsadmin.quota()`, `dsadmin.quota_delete()`, `dsadmin.quota_exists()`, `dsadmin.quota_save()`, `dsadmin.quota_usage()`, `dsadmin.quotas()`, `opal.quotas()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.quota_enable(o, enabled = FALSE, type = 'user', subject = 'andrei')
opal.logout(o)

## End(Not run)
```

`dsadmin.quota_exists` *Check a DataSHIELD usage quota exists*

Description

Check whether a DataSHIELD usage quota was defined for a subject and a usage metric.

Usage

```
dsadmin.quota_exists(
  opal,
  metric = "EXECUTION_TIME",
  type = "system",
  subject = NULL
)
```

Arguments

`opal`	Opal object.
`metric`	The usage metric: 'EXECUTION_TIME' (default) or 'SESSION_TIME'.
`type`	The type of subject: system (default), user or group.
`subject`	The user name or the group name, not applicable when the type is system.

Details

Requires DataSHIELD or system administration permissions.

See Also

Other DataSHIELD quotas: `dsadmin.quota()`, `dsadmin.quota_delete()`, `dsadmin.quota_enable()`, `dsadmin.quota_save()`, `dsadmin.quota_usage()`, `dsadmin.quotas()`, `opal.quotas()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
if (!dsadmin.quota_exists(o, type = 'group', subject = 'datashield'))
  dsadmin.quota_save(o, limit = 60, type = 'group', subject = 'datashield')
opal.logout(o)

## End(Not run)
```

`dsadmin.quota_save` *Save a DataSHIELD usage quota*

Description

Create or update the DataSHIELD usage quota of a subject: the system default, a group or a user. A subject can have one quota per usage metric, therefore saving replaces the quota that the subject already had for this metric, if any.

Usage

```
dsadmin.quota_save(
  opal,
  limit,
  period = "WEEKLY",
  metric = "EXECUTION_TIME",
  type = "system",
  subject = NULL,
  enabled = TRUE
)
```

Arguments

`opal`	Opal object.
`limit`	The allowance, in minutes of whatever the metric names. Zero is a meaningful value: it forbids DataSHIELD altogether for the subject.
`period`	The rolling window the usage is summed over: 'WEEKLY' (default) or 'DAILY'. There is no reset instant, capacity returns as old activity ages out of the window.
`metric`	The usage metric: 'EXECUTION_TIME' (default) is the R server's cpu time consumed (an idle session costs nothing), 'SESSION_TIME' is the time R sessions are held, whether they compute or not.
`type`	The type of subject: system (default), user or group.
`subject`	The user name or the group name, not applicable when the type is system.
`enabled`	Whether the quota applies (default is TRUE).

Details

When a user runs DataSHIELD, the quota that applies to them is their own if they have one, else the most permissive of the ones given to the groups they belong to, else the system default. A disabled quota is ignored by this resolution.

Requires DataSHIELD or system administration permissions.

Value

The saved DataSHIELD quota, invisibly.

See Also

Other DataSHIELD quotas: dsadmin.quota(), dsadmin.quota_delete(), dsadmin.quota_enable(), dsadmin.quota_exists(), dsadmin.quota_usage(), dsadmin.quotas(), opal.quotas()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# 10 hours of R execution time per week for everyone
dsadmin.quota_save(o, limit = 600)
# 2 hours of R execution time per day for a user
dsadmin.quota_save(o, limit = 120, period = 'DAILY', type = 'user', subject = 'andrei')
# 8 hours of R session time per week for a group
dsadmin.quota_save(o, limit = 480, metric = 'SESSION_TIME', type = 'group', subject = 'datashield')
dsadmin.quotas(o)
opal.logout(o)

## End(Not run)
```

---

dsadmin.quota_usage *Get the DataSHIELD usage quota of a user*

Description

Get, for each usage metric, the DataSHIELD quota that applies to a user and what they have consumed against it. Every metric is reported, whether or not a quota applies to it: an entry with no limit is the unlimited case, and nothing is measured then.

Usage

```
dsadmin.quota_usage(opal, user, df = TRUE)
```

Arguments

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

Details

Requires DataSHIELD or system administration permissions. See opal.quotas for the quotas of the current user, which requires no permission.

Value

The DataSHIELD quota usages as a data.frame (columns are: context, user, metric, period, limit, used, exceeded, windowStartDate, nextCreditDate, openSessionsCount) or a list. Limits and usages are expressed in minutes, and are NA when no quota applies to the metric.

See Also

Other DataSHIELD quotas: dsadmin.quota(), dsadmin.quota_delete(), dsadmin.quota_enable(), dsadmin.quota_exists(), dsadmin.quota_save(), dsadmin.quotas(), opal.quotas()

Examples

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

## End(Not run)
```

---

dsadmin.remove_package

Remove DataSHIELD package

Description

Remove a DataSHIELD package permanently.

Usage

```
dsadmin.remove_package(opal, pkg, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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", profile = NULL)
```

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"
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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, profile = 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).
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
name	Name of the option
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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_options *Remove all DataSHIELD options*

Description

Remove all DataSHIELD options

Usage

```
dsadmin.rm_options(opal, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
dsadmin.rm_options(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, profile = 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).
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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",
  profile = NULL
)
```

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 or function code.
`path`	Path to the R file containing the script (mutually exclusive with func).
`type`	Type of the method: "aggregate" (default) or "assign"
`profile`	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# as a package's function
dsadmin.set_method(o, 'foo', func = 'base::mean')
# as a custom function
dsadmin.set_method(o, 'foo', func = function(x) { base::mean(x) })
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, profile = NULL)
```

Arguments

`opal`	Opal object or list of opal objects.
`name`	Name of the option
`value`	Value of the option
`profile`	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

See Also

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

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, profile = 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).
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

Value

TRUE if successfull

See Also

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

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)
```

dsadmin.unpublish_package *Unpublish DataSHIELD package settings*

Description

Remove DataSHIELD aggregate/assign methods and options as defined by the package from the DataSHIELD configuration.

Usage

```
dsadmin.unpublish_package(opal, pkg, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The DataSHIELD profile name to which operation applies. See also dsadmin.profiles.

Value

TRUE if successfull

See Also

Other DataSHIELD functions: dsadmin.get_method(), dsadmin.get_methods(), dsadmin.get_options(), dsadmin.install_github_package(), dsadmin.install_local_package(), dsadmin.install_package(), dsadmin.installed_package(), dsadmin.package_description(), dsadmin.package_descriptions(), dsadmin.publish_package(), dsadmin.remove_package(), dsadmin.rm_method(), dsadmin.rm_methods(), dsadmin.rm_option(), dsadmin.rm_options(), 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.unpublish_package(o, 'dsBase')
opal.logout(o)

## End(Not run)
```

`oadmin.activity` *Get R activity*

Description

Get the recorded R session metrics.

Usage

```
oadmin.activity(
  opal,
  user = NULL,
  profile = NULL,
  from = NULL,
  to = NULL,
  df = TRUE
)
```

Arguments

`opal`	Opal connection object.
`user`	Optional user name.
`profile`	Optional profile name.
`from`	Optional start date.
`to`	Optional end date.
`df`	Return a data.frame (default is TRUE)

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# all sessions metrics
oadmin.activity(o)
# user and profile specific session metrics
oadmin.activity(o, user = 'dsuser', profile = 'default')
# user sessions in a time range
oadmin.activity(o, user = "dsuser", from = "2022-07-01", to = "2023-01-01")
opal.logout(o)

## End(Not run)
```

`oadmin.activity_summary` *Get R activity summary*

Description

Get the recorded R session metrics, grouped by profile and user.

Usage

```
oadmin.activity_summary(
  opal,
  user = NULL,
  profile = NULL,
  from = NULL,
  to = NULL,
  df = TRUE
)
```

Arguments

`opal`	Opal connection object.
`user`	Optional user name.
`profile`	Optional profile name.
`from`	Optional start date.
`to`	Optional end date.
`df`	Return a data.frame (default is TRUE)

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# all sessions metrics
oadmin.activity_summary(o)
# user and profile specific session metrics
oadmin.activity_summary(o, user = 'dsuser', profile = 'default')
# user sessions in a time range
oadmin.activity_summary(o, user = "dsuser", from = "2022-07-01", to = "2023-01-01")
opal.logout(o)

## End(Not run)
```

oadmin.installed_devtools

Check devtools package

Description

Check if devtools package is installed.

Usage

```
oadmin.installed_devtools(opal, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
profile	The R servers profile name to which operation applies. See also opal.profiles.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_cran_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_package(), 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_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, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The R servers profile name to which operation applies. See also opal.profiles.

Value

TRUE if installed

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_cran_package(), oadmin.install_devtools(), oadmin.install_github_package(), 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

Get the installed packages from all the R servers in the cluster described by the profile.

Usage

```
oadmin.installed_packages(opal, profile = NULL)
```

Arguments

opal	Opal object.
profile	The R servers profile name to which operation applies. See also opal.profiles.

Value

The result of the installed.packages() call

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_cran_package(), oadmin.install_devtools(), oadmin.install_github_package(), 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, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The R servers profile name to which operation applies. See also opal.profiles.

See Also

Other administration functions: oadmin.install_cran_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_package(), oadmin.installed_packages(), 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_cran_package

Install a package from CRAN

Description

Install a package from configured CRAN repositories.

Usage

```
oadmin.install_cran_package(opal, pkg, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The R servers profile name to which operation applies. See also opal.profiles.

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_package(), oadmin.installed_packages(), oadmin.package_description(), oadmin.remove_package()

Examples

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

Arguments

opal	Opal object or list of opal objects.
profile	The R servers profile name to which operation applies. See also opal.profiles.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_cran_package(), oadmin.install_github_package(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_package(), oadmin.installed_packages(), 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_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",
  profile = 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".
profile	The R servers profile name to which operation applies. See also opal.profiles.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_cran_package(), oadmin.install_devtools(), oadmin.install_local_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_package(), oadmin.installed_packages(), 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. The R server profile to which the operation applies is the one specified at login time.

Usage

```
oadmin.install_local_package(opal, path, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
path	Path to the package archive file.
profile	The R servers profile name to which operation applies. See also opal.profiles.

See Also

Other administration functions: oadmin.install_bioconductor_package(), oadmin.install_cran_package(), oadmin.install_devtools(), oadmin.install_github_package(), oadmin.install_package(), oadmin.installed_devtools(), oadmin.installed_package(), oadmin.installed_packages(), 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 CRAN package*

Description

Install package from CRAN repos. To install the latest version of a package, it has to be removed first.

Usage

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

Arguments

`opal`	Opal object or list of opal objects.
`pkg`	Package name.
`repos`	Character vector, the base URLs of the repositories to use.
`profile`	The R servers profile name to which operation applies. See also opal.profiles.

Value

TRUE if successfully installed

See Also

Other administration functions: `oadmin.install_bioconductor_package()`, `oadmin.install_cran_package()`, `oadmin.install_devtools()`, `oadmin.install_github_package()`, `oadmin.install_local_package()`, `oadmin.installed_devtools()`, `oadmin.installed_package()`, `oadmin.installed_packages()`, `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.log *Get Opal main logs*

Description

Get Opal main logs

Usage

```
oadmin.log(opal, all = TRUE)
```

Arguments

opal	Opal connection object.
all	Get all or only latest log messages.

Examples

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

## End(Not run)
```

---

oadmin.log_rest *Get Opal REST API logs*

Description

Get Opal REST API logs

Usage

```
oadmin.log_rest(opal, all = TRUE)
```

Arguments

opal	Opal connection object.
all	Get all or only latest log messages.

Examples

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

## End(Not run)
```

---

oadmin.log_sql *Get Opal SQL API logs*

Description

Get Opal SQL API logs

Usage

```
oadmin.log_sql(opal, all = TRUE)
```

Arguments

opal	Opal connection object.
all	Get all or only latest log messages.

Examples

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

## End(Not run)
```

`oadmin.package_description` *Get package description*

Description

Get package description

Usage

```
oadmin.package_description(opal, pkg, fields = NULL, profile = 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.
`profile`	The R servers profile name to which operation applies. See also opal.profiles.

See Also

Other administration functions: `oadmin.install_bioconductor_package()`, `oadmin.install_cran_package()`, `oadmin.install_devtools()`, `oadmin.install_github_package()`, `oadmin.install_local_package()`, `oadmin.install_package()`, `oadmin.installed_devtools()`, `oadmin.installed_package()`, `oadmin.installed_packages()`, `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 (deprecated)*

Description

Deprecated, use oadmin.r_perm.

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.r_perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.r_perm(o)
oadmin.r_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

oadmin.perm_add *Add or update a R permission (deprecated)*

Description

Deprecated, use oadmin.r_perm_add.

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.r_perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.r_perm(o)
oadmin.r_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

`oadmin.perm_delete` *Delete a R permission (deprecated)*

Description

Deprecated, use oadmin.r_perm_delete.

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.r_perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.r_perm(o)
oadmin.r_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, profile = NULL)
```

Arguments

opal	Opal object or list of opal objects.
pkg	Package name.
profile	The R servers profile name to which operation applies. See also opal.profiles.

See Also

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

oadmin.r_perm *Get the R permissions*

Description

Get the permissions that were applied to the R service.

Usage

```
oadmin.r_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.r_perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.r_perm(o)
oadmin.r_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

---

oadmin.r_perm_add *Add or update a R permission*

Description

Add or update a permission on the R service.

Usage

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

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.r_perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.r_perm(o)
oadmin.r_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

oadmin.r_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.r_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.r_perm_add(o, c('andrei', 'valentina'), 'user', 'use')
oadmin.r_perm(o)
oadmin.r_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

oadmin.system_metrics *Get system metrics*

Description

Get some metrics about the Opal system status. The following information are returned: ‘timestamp‘ (the EPOC time at which the metrics were collected), ‘uptime‘ (the running time in millis), ‘heapMemory‘ (the memory currently used), ‘nonHeapMemory‘ (the memory that can be used), ‘threads‘ (the current (count) and maximum (peak) numbers of threads), ‘gcs‘ (the garbage collectors activity).

Usage

```
oadmin.system_metrics(opal)
```

Arguments

opal	Opal connection object.

Examples

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

## End(Not run)
```

---

oadmin.system_perm *Get the System permissions*

Description

Get the permissions that were applied to the whole system.

Usage

```
oadmin.system_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.system_perm_add(o, c('andrei', 'valentina'), 'user', 'project_add')
oadmin.system_perm(o)
oadmin.system_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

`oadmin.system_perm_add`

Add or update a System permission

Description

Add or update a permission on the whole system.

Usage

```
oadmin.system_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: project_add, audit or administrate.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.system_perm_add(o, c('andrei', 'valentina'), 'user', 'project_add')
oadmin.system_perm(o)
oadmin.system_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

`oadmin.system_perm_delete`

Delete a System permission

Description

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

Usage

```
oadmin.system_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.system_perm_add(o, c('andrei', 'valentina'), 'user', 'project_add')
oadmin.system_perm(o)
oadmin.system_perm_delete(o, c('andrei', 'valentina'), 'user')
opal.logout(o)

## End(Not run)
```

---

oadmin.users *Get the users*

Description

Get the users from the Opal internal users registry.

Usage

```
oadmin.users(opal, df = TRUE)
```

Arguments

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

See Also

Other user functions: oadmin.user_add(), oadmin.user_delete(), oadmin.user_enable(), oadmin.user_exists(), oadmin.user_profile_delete(), oadmin.user_profiles(), oadmin.user_reset_password(

Examples

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

## End(Not run)
```

oadmin.user_add *Add a user*

Description

Add a user in Opal internal users registry.

Usage

```
oadmin.user_add(opal, name, groups = NULL, password = NULL)
```

Arguments

opal	Opal object.
name	User name
groups	User groups
password	User password. If not provided, a password will be generated and returned.

See Also

Other user functions: oadmin.user_delete(), oadmin.user_enable(), oadmin.user_exists(), oadmin.user_profile_delete(), oadmin.user_profiles(), oadmin.user_reset_password(), oadmin.users()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
pwd <- oadmin.user_add(o, "foo", groups = c("datashield", "CNSIM"))
opal.logout(o)

## End(Not run)
```

oadmin.user_delete *Delete a user*

Description

Delete a user from Opal internal users registry. Fails silently if user does not exist.

Usage

```
oadmin.user_delete(opal, name)
```

Arguments

opal	Opal object.
name	User name

See Also

Other user functions: oadmin.user_add(), oadmin.user_enable(), oadmin.user_exists(), oadmin.user_profile_delete(), oadmin.user_profiles(), oadmin.user_reset_password(), oadmin.users()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
pwd <- oadmin.user_add(o, "foo", groups = c("datashield", "CNSIM"))
oadmin.user_delete(o, "foo")
opal.logout(o)

## End(Not run)
```

---

oadmin.user_enable *Enable a user*

Description

Enable or disable a user from Opal internal users registry.

Usage

```
oadmin.user_enable(opal, name, enabled = TRUE)
```

Arguments

opal	Opal object.
name	User name
enabled	Logical to enable a user.

See Also

Other user functions: oadmin.user_add(), oadmin.user_delete(), oadmin.user_exists(), oadmin.user_profile_delete(), oadmin.user_profiles(), oadmin.user_reset_password(), oadmin.users()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
pwd <- oadmin.user_add(o, "foo", groups = c("datashield", "CNSIM"))
oadmin.user_enable(o, "foo", enabled = FALSE)
opal.logout(o)

## End(Not run)
```

`oadmin.user_exists` *Check user exists*

Description

Check whether a user exists, either in the internal user registry (see oadmin.users) or as an external user that already logged in (see oadmin.user_profiles).

Usage

```
oadmin.user_exists(opal, name)
```

Arguments

`opal`	Opal object.
`name`	User name

See Also

Other user functions: `oadmin.user_add()`, `oadmin.user_delete()`, `oadmin.user_enable()`, `oadmin.user_profile_delete()`, `oadmin.user_profiles()`, `oadmin.user_reset_password()`, `oadmin.users()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
if (!oadmin.user_exists(o, "foo"))
  oadmin.user_add(o, "foo", password = "bar123")
opal.logout(o)

## End(Not run)
```

oadmin.user_profiles *Get user profiles*

Description

When a user has logged in Opal, he/she has a profile representing its activity. The user can be defined in the Opal internal user registry, or in an external realm.

Usage

```
oadmin.user_profiles(opal, df = TRUE)
```

Arguments

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

See Also

Other user functions: oadmin.user_add(), oadmin.user_delete(), oadmin.user_enable(), oadmin.user_exists(), oadmin.user_profile_delete(), oadmin.user_reset_password(), oadmin.users()

Examples

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

## End(Not run)
```

oadmin.user_profile_delete *Delete a user profile*

Description

Delete a user profile without deleting user if this one is defined in the Opal internal users registry. Fails silently if user profile does not exist. A user profile is the footprint of a user, created at first login. It keeps track of its activity, the realm from which he/she was authenticated, its groups at time of the last login and more.

Usage

```
oadmin.user_profile_delete(opal, name)
```

Arguments

opal	Opal object.
name	User name

See Also

Other user functions: oadmin.user_add(), oadmin.user_delete(), oadmin.user_enable(), oadmin.user_exists(), oadmin.user_profiles(), oadmin.user_reset_password(), oadmin.users()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
pwd <- oadmin.user_add(o, "foo", groups = c("datashield", "CNSIM"))
oadmin.user_profile_delete(o, "foo")
opal.logout(o)

## End(Not run)
```

---

oadmin.user_reset_password

Reset user password

Description

Reset the password of a user from Opal internal users registry.

Usage

```
oadmin.user_reset_password(opal, name, password = NULL)
```

Arguments

opal	Opal object.
name	User name
password	User password. If not provided, a password will be generated and returned.

See Also

Other user functions: oadmin.user_add(), oadmin.user_delete(), oadmin.user_enable(), oadmin.user_exists(), oadmin.user_profile_delete(), oadmin.user_profiles(), oadmin.users()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
pwd <- oadmin.user_add(o, "foo", groups = c("datashield", "CNSIM"))
oadmin.user_reset_password(o, "foo", password = "password1234")
oadmin.user_rm(o, "foo")
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.datasource()`, `opal.datasources()`, `opal.table()`, `opal.tables()`, `opal.valueset()`, `opal.variable()`, `opal.variable_summary()`, `opal.variables()`

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.datasource()`, `opal.datasources()`, `opal.table()`, `opal.tables()`, `opal.valueset()`, `opal.variable()`, `opal.variable_summary()`, `opal.variables()`

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(), opal.assign.table.tibble()

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(), opal.assign.resource(), opal.assign.script(), opal.assign.table(), opal.assign.table.tibble()

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(), opal.assign.data(), opal.assign.script(), opal.assign.table(), opal.assign.table.tibble()

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(), opal.assign.data(), opal.assign.resource(), opal.assign.table(), opal.assign.table.tibble()

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(), opal.assign.data(), opal.assign.resource(), opal.assign.script(), opal.assign.table.tibble()

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(), opal.assign.data(), opal.assign.resource(), opal.assign.script(), opal.assign.table()

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 attribute values*

Description

Get a vector of attribute 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.datasource(), opal.datasources(), opal.table(), opal.tables(), opal.valueset(), opal.variable(), opal.variable_summary(), opal.variables()

Examples

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

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(), opal.command_result(), opal.command_rm(), opal.commands_rm()

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(), opal.command_result(), opal.command_rm(), opal.commands()

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()`, `opal.command_rm()`, `opal.commands()`, `opal.commands_rm()`

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(), opal.command_result(), opal.commands(), opal.commands_rm()

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.table(), opal.tables(), opal.valueset(), opal.variable(), opal.variable_summary(), opal.variables()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.datasource(o, 'CNSIM')
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.table(), opal.tables(), opal.valueset(), opal.variable(), opal.variable_summary(), opal.variables()

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_info()`, `opal.file_ls()`, `opal.file_mkdir()`, `opal.file_mkdir_tmp()`, `opal.file_mv()`, `opal.file_read()`, `opal.file_rm()`, `opal.file_unzip()`, `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(), opal.file_download(), opal.file_info(), opal.file_ls(), opal.file_mkdir(), opal.file_mkdir_tmp(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_unzip(), opal.file_upload(), opal.file_write()

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, optionally encrypted with provided key. If the file is a folder and/or it is to be encrypted, the resulting downloaded file will be a zip archive file. The process of preparing this file bundle will be done in a background task. Then the result of this task, if successful, will be downloaded.

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 omitted, 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(), opal.file_cp(), opal.file_info(), opal.file_ls(), opal.file_mkdir(), opal.file_mkdir_tmp(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_unzip(), opal.file_upload(), opal.file_write()

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_info *Get file information*

Description

Get details about a file from the Opal file system.

Usage

```
opal.file_info(opal, path)
```

Arguments

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

See Also

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

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.file_info(o, '/home/administrator/joins/join-src-3.csv')
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(), opal.file_cp(), opal.file_download(), opal.file_info(), opal.file_mkdir(), opal.file_mkdir_tmp(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_unzip(), opal.file_upload(), opal.file_write()

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()`, `opal.file_cp()`, `opal.file_download()`, `opal.file_info()`, `opal.file_ls()`, `opal.file_mkdir_tmp()`, `opal.file_mv()`, `opal.file_read()`, `opal.file_rm()`, `opal.file_unzip()`, `opal.file_upload()`, `opal.file_write()`

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 user personal 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(), opal.file_cp(), opal.file_download(), opal.file_info(), opal.file_ls(), opal.file_mkdir(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_unzip(), opal.file_upload(), opal.file_write()

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(), opal.file_cp(), opal.file_download(), opal.file_info(), opal.file_ls(), opal.file_mkdir(), opal.file_mkdir_tmp(), opal.file_read(), opal.file_rm(), opal.file_unzip(), opal.file_upload(), opal.file_write()

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(), opal.file_cp(), opal.file_download(), opal.file_info(), opal.file_ls(), opal.file_mkdir(), opal.file_mkdir_tmp(), opal.file_mv(), opal.file_rm(), opal.file_unzip(), opal.file_upload(), opal.file_write()

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(), opal.file_cp(), opal.file_download(), opal.file_info(), opal.file_ls(), opal.file_mkdir(), opal.file_mkdir_tmp(), opal.file_mv(), opal.file_read(), opal.file_unzip(), opal.file_upload(), opal.file_write()

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_unzip` *Unzip a zip archive file*

Description

Unzip a zip archive file from the Opal file system.

Usage

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

Arguments

`opal`	Opal object.
`source`	Path to the file in the Opal file system (must exist and have the ".zip" file extension).
`destination`	Path to the destination file or folder in the Opal file system.
`key`	Key to decrypt archive.

Value

The path of the extracted archive folder in the Opal file system.

See Also

Other file functions: `opal.file()`, `opal.file_cp()`, `opal.file_download()`, `opal.file_info()`, `opal.file_ls()`, `opal.file_mkdir()`, `opal.file_mkdir_tmp()`, `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')
# unzip
path <- opal.file_unzip(o, "/tmp/TESTING.zip", "/home/administrator")
opal.logout(o)

## End(Not run)
```

opal.file_upload *Upload a file or a folder*

Description

Upload a file or a folder into the Opal file system. Creates the destination folder (and parents) when necessary. Hidden files and folders (with name starting with dot) can be excluded.

Usage

```
opal.file_upload(opal, source, destination, all.files = TRUE)
```

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. Folder (and parents) will be created if missing.
all.files	When FALSE, upload only visible files (following Unix-style visibility, that is files whose name does not start with a dot). Default is TRUE.

See Also

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

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# upload a file
opal.file_upload(o, 'some_data.csv', '/home/administrator')
# upload a folder
opal.file_upload(o, 'some_data', '/home/administrator')
opal.logout(o)

## End(Not run)
```

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(), opal.file_cp(), opal.file_download(), opal.file_info(), opal.file_ls(), opal.file_mkdir(), opal.file_mkdir_tmp(), opal.file_mv(), opal.file_read(), opal.file_rm(), opal.file_unzip(), opal.file_upload()

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(),
  acceptType = "application/json",
  outFile = NULL,
  callback = NULL
)
```

Arguments

opal	Opal object.
...	Resource path segments.
query	Named list of query parameters.
acceptType	The type of the body content. Default is 'application/json', i.e. a serialized R object or an error message.
outFile	Write response body to file. Ignored if NULL (default).
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', 'CNSIM')
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(), opal.execute.source(), 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). Different login strategies are possible: (1) by providing username/password, or (2) by providing username/password and a one-time password code (TOPT) when user has activated two-factor authentication, or (3) by providing a personal access token (PAT), or (4) by providing a key pair in PEM format.

Usage

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

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. Secure http (https) connection is required.
opts	Curl options as described by httr (call httr::httr_options() for details). Can be provided by "opal.opts" option.
profile	R server profile name. This will drive the R server in which a R session will be created. If no remote R session is needed (because Opal specific operations are done), this parameter does not need to be provided. Otherwise, if missing, the default R server profile will be applied ('default'). See also opal.profiles. Can be provided by "opal.profile" option.
restore	Workspace ID to be restored (see also opal.logout)
context	Context of the R session to be created. Either "r" (default) or "datashield".

Value

A opal object or a list of opal objects.

See Also

Other connection functions: `opal.logout()`, `opal.profiles()`

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)

# login with a R server profile
o <- opal.login(username = 'administrator', password = 'password',
                url = 'https://opal-demo.obiba.org', profile = 'default')
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(), opal.profiles()

Examples

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

## End(Not run)
```

---

opal.perms *Get the permissions of a subject*

Description

Get the permissions of a subject. If the subject is a user, the permissions of the groups to which the user belongs are also added to the result.

Usage

```
opal.perms(opal, subject, type = "user")
```

Arguments

opal	Opal connection object.
subject	A subject identifier: user or group name.
type	The type of subject: user (default) or group.

Value

A data.frame with columns: subject, type, target (path to the opal object to which the permission applies), target_type and perm (the permission name)

See Also

Other table functions: opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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.perms(o, 'andrei')
opal.table_perm_delete(o, 'CNSIM', 'CNSIM1', c('andrei', 'valentina'), 'user')
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",
  acceptType = "application/json",
  outFile = NULL,
  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. Default is 'application/x-rscript'.
`acceptType`	The type of the body content. Default is 'application/json', i.e. a serialized R object or an error message.
`outFile`	Write response body to file. Ignored if NULL (default).
`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.profiles *List R profiles*

Description

Each R profile corresponds one R servers cluster name. These profiles names can be provided when login (see opal.login) and on some package administration operations.

Usage

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

Arguments

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

Value

The R profiles as a data.frame or a list

See Also

Other connection functions: opal.login(), opal.logout()

Examples

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

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(), opal.project_create(), opal.project_delete(), opal.project_exists(), opal.projects_databases()

Examples

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

## End(Not run)
```

---

opal.projects_databases *Get projects databases*

Description

When creating a project for storing data, it is required to name the database to be associated.

Usage

```
opal.projects_databases(opal)
```

Arguments

opal	Opal object.

Value

A character vector of databases names.

See Also

Other project functions: opal.project(), opal.project_create(), opal.project_delete(), opal.project_exists(), opal.projects()

Examples

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

## End(Not run)
```

`opal.project_backup` *Backup a project*

Description

The project backup task has a limited scope: tables (dictionary and data export), views (either as a logical table or as an exported table), resources, files and report templates. Other project elements that are not part of the backup: user and group permissions, view change history, table analysis, report executions etc.

Usage

```
opal.project_backup(
  opal,
  project,
  archive,
  viewsAsTables = FALSE,
  override = TRUE,
  wait = TRUE
)
```

Arguments

`opal`	Opal object.
`project`	Name of the project.
`archive`	Archive directory path in the Opal file system. If folder (and parents) does not exist, it will be created.
`viewsAsTables`	Treat views as tables, i.e. export data instead of keeping derivation scripts. Default is FALSE.
`override`	Overwrite an existing backup folder. Default is TRUE.
`wait`	Wait for backup task completion. Default is TRUE.

Value

The project command ID if wait parameter is FALSE. See opal.project_command to retrieve asynchronous command state.

Examples

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

## End(Not run)
```

opal.project_command *Get project task*

Description

Get the project's task command object.

Usage

```
opal.project_command(opal, project, id)
```

Arguments

opal	Opal object.
project	Name of the project.
id	The project command ID.

Value

The command state object.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
id <- opal.project_backup(o, 'GREENSPACE', '/home/administrator/backup/GREENSPACE', wait = FALSE)
opal.project_command(opal, 'GREENSPACE', id)
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,
  internal = FALSE
)
```

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. If the value is a logical and is TRUE, first the internal flag will be evaluated, otherwise the default database will be selected or the first one if there is no default.
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).
internal	When database is TRUE and this flag is TRUE as well, the project will be associated to an internal database (Opal 6.0+ only).

See Also

Other project functions: opal.project(), opal.project_delete(), opal.project_exists(), opal.projects(), opal.projects_databases()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# with named database
opal.project_create(o, 'test', database='opal_data', title='This is a test', tags=list('Test'))
# with internal database
opal.project_create(o, 'test_internal_db', database = TRUE, internal = TRUE)
# with default database
opal.project_create(o, 'test_default_db', database = TRUE)
# no database, for views and resources only
opal.project_create(o, 'test_no_db')
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(), opal.project_create(), opal.project_exists(), opal.projects(), opal.projects_databases()

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(), opal.project_create(), opal.project_delete(), opal.projects(), opal.projects_databases()

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 = "administrate"
)
```

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.project_restore *Restore a project*

Description

Restore the data of a project from a backup archive file to be found on the Opal file system. The destination project must exist and can have a name different from the original one (beware that this could break views). Default behavior is to stop when an item to restore already exist (override can be forced).

Usage

```
opal.project_restore(
  opal,
  project,
  archive,
  key = NULL,
  override = TRUE,
  wait = TRUE
)
```

Arguments

opal	Opal object.
project	Name of the project.
archive	Archive directory or zip file path in the Opal file system.
key	Archive zip file password (if applies).
override	Overwrite existing items (table, view, resource, report). Project files override is not checked. Default is TRUE.
wait	Wait for restore task completion. Default is TRUE.

Value

The project command ID if wait parameter is FALSE. See opal.project_command to retrieve asynchronous command state.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# create the project to restore, with the default database (to store tables)
opal.project_create(o, 'GREENSPACE2', database = TRUE)
# upload backup zip and launch restore task
opal.file_upload(o, 'GREENSPACE.zip', '/home/administrator')
opal.project_restore(o, 'GREENSPACE2', '/home/administrator/GREENSPACE.zip')
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.quotas *Get own R usage quotas*

Description

Get, for each usage metric, the R usage quota that applies to the current user in an execution context and what they have consumed against it. Every metric is reported, whether or not a quota applies to it: an entry with no limit is the unlimited case, and nothing is measured then.

Usage

```
opal.quotas(opal, context = "DataSHIELD", df = TRUE)
```

Arguments

opal	Opal object.
context	The R execution context: 'DataSHIELD' (default), 'R', 'Import', 'Export', 'SQL' or 'Analyse'.
df	Return a data.frame (default is TRUE)

Details

Reporting on oneself requires no special permission. See dsadmin.quota_usage for the quotas of another user in the DataSHIELD context.

Value

The R quota usages as a data.frame (columns are: context, user, metric, period, limit, used, exceeded, windowStartDate, nextCreditDate, openSessionsCount) or a list. Limits and usages are expressed in minutes, and are NA when no quota applies to the metric.

See Also

Other DataSHIELD quotas: dsadmin.quota(), dsadmin.quota_delete(), dsadmin.quota_enable(), dsadmin.quota_exists(), dsadmin.quota_save(), dsadmin.quota_usage(), dsadmin.quotas()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.quotas(o)
opal.quotas(o, context = 'R')
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 resource functions: opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources(), opal.resources_perm(), opal.resources_perm_add(), opal.resources_perm_delete()

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 resource functions: opal.resource(), opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources_perm(), opal.resources_perm_add(), opal.resources_perm_delete()

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

See Also

Other resource functions: opal.resource(), opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources(), opal.resources_perm_add(), opal.resources_perm_delete()

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.

See Also

Other resource functions: opal.resource(), opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources(), opal.resources_perm(), opal.resources_perm_delete()

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.

See Also

Other resource functions: opal.resource(), opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources(), opal.resources_perm(), opal.resources_perm_add()

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 resource functions: opal.resource(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources(), opal.resources_perm(), opal.resources_perm_add(), opal.resources_perm_delete()

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.

See Also

Other resource functions: opal.resource(), opal.resource_create(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources(), opal.resources_perm(), opal.resources_perm_add(), opal.resources_perm_delete()

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 resource functions: `opal.resource()`, `opal.resource_create()`, `opal.resource_delete()`, `opal.resource_extension_create()`, `opal.resource_get()`, `opal.resource_perm()`, `opal.resource_perm_add()`, `opal.resource_perm_delete()`, `opal.resources()`, `opal.resources_perm()`, `opal.resources_perm_add()`, `opal.resources_perm_delete()`

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_extension_create` *Create an extended resource reference in a project*

Description

Create an extended resource reference in a project

Usage

```
opal.resource_extension_create(
  opal,
  project,
  name,
  provider,
  factory,
  parameters,
  description = NULL,
  credentials = NULL
)
```

Arguments

`opal`	Opal object.
`project`	Name of the project.
`name`	Name of the resource in the project.
`provider`	Name of the R package in which the resource is defined.
`factory`	Name of the JS function that turns parameters and credentials into a resource object.
`parameters`	A named list of the resource parameters.
`description`	The description of the resource (optional).
`credentials`	A named list of the resource credentials (optional).

See Also

Other resource functions: `opal.resource()`, `opal.resource_create()`, `opal.resource_delete()`, `opal.resource_exists()`, `opal.resource_get()`, `opal.resource_perm()`, `opal.resource_perm_add()`, `opal.resource_perm_delete()`, `opal.resources()`, `opal.resources_perm()`, `opal.resources_perm_add()`, `opal.resources_perm_delete()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_extension_create(o, 'RSRC', 'ga4gh_1000g',
  provider = 'dsOmics', factory = 'ga4gh-htsget',
  parameters = list(
    host = 'https://htsget.ga4gh.org',
    sample = '1000genomes.phase1.chr1',
    reference = '1',
    start = '1',
    end = '100000',
    format = 'GA4GHVCF'
  )
)
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 resource functions: opal.resource(), opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_perm(), opal.resource_perm_add(opal.resource_perm_delete(), opal.resources(), opal.resources_perm(), opal.resources_perm_add(), opal.resources_perm_delete()

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

See Also

Other resource functions: opal.resource(), opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm_add(), opal.resource_perm_delete(), opal.resources(), opal.resources_perm(), opal.resources_perm_add(), opal.resources_perm_delete()

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.

See Also

Other resource functions: `opal.resource()`, `opal.resource_create()`, `opal.resource_delete()`, `opal.resource_exists()`, `opal.resource_extension_create()`, `opal.resource_get()`, `opal.resource_perm()`, `opal.resource_perm_delete()`, `opal.resources()`, `opal.resources_perm()`, `opal.resources_perm_add()`, `opal.resources_perm_delete()`

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.

See Also

Other resource functions: opal.resource(), opal.resource_create(), opal.resource_delete(), opal.resource_exists(), opal.resource_extension_create(), opal.resource_get(), opal.resource_perm(), opal.resource_perm_add(), opal.resources(), opal.resources_perm(), opal.resources_perm_add(), opal.resources_perm_delete()

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_view_create` *Create an Opal view over a resource reference*

Description

Create an Opal view if a table with same name does not already exist. The resource reference is required.The dictionary of the created view will be discovered at initialization time. Use opal.table_dictionary_update to apply a dictionary.

Usage

```
opal.resource_view_create(
  opal,
  project,
  table,
  resource,
  type = "Participant",
  idColumn = NULL,
  profile = NULL
)
```

Arguments

`opal`	Opal connection object.
`project`	Project name where the view will be located.
`table`	View name to be created.
`resource`	Fully qualified resource name.
`type`	Entity type, default is "Participant".
`idColumn`	Name of the column which contains the entity identifiers. If not specified, the first column will be used.
`profile`	R server profile to use for establishing the connection with the resource. If not specifed, the profile will guessed based on the resource definition.

See Also

Other table functions: `opal.perms()`, `opal.resource_view_reconnect()`, `opal.table_create()`, `opal.table_delete()`, `opal.table_dictionary_get()`, `opal.table_dictionary_update()`, `opal.table_exists()`, `opal.table_export()`, `opal.table_get()`, `opal.table_import()`, `opal.table_perm()`, `opal.table_perm_add()`, `opal.table_perm_delete()`, `opal.table_save()`, `opal.table_truncate()`, `opal.table_view_create()`, `opal.table_view_update()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# make a view over a resource
opal.resource_view_create(o, "CNSIM", "CNSIM4", resource = "RSRC.CNSIM1")
opal.resource_view_create(o, "CNSIM", "FEMALE_2439",
                          resource = "RSRC.FEMALE_2439", idColumn = "Name")
opal.logout(o)

## End(Not run)
```

`opal.resource_view_reconnect`

Reconnect an Opal view to its underlying resource

Description

A view over a resource handles a connection to this resource. When the resource changes (data update, broken connection etc.), the connection to this resource can be re-initialized.

Usage

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

Arguments

`opal`	Opal connection object.
`project`	Project name where the view is located.
`table`	View name to be reconnected.

See Also

Other table functions: `opal.perms()`, `opal.resource_view_create()`, `opal.table_create()`, `opal.table_delete()`, `opal.table_dictionary_get()`, `opal.table_dictionary_update()`, `opal.table_exists()`, `opal.table_export()`, `opal.table_get()`, `opal.table_import()`, `opal.table_perm()`, `opal.table_perm_add()`, `opal.table_perm_delete()`, `opal.table_save()`, `opal.table_truncate()`, `opal.table_view_create()`, `opal.table_view_update()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# make a view over a resource
opal.resource_view_create(o, "CNSIM", "CNSIM4", resource = "RSRC.CNSIM1")
# re-initialize the view's connection to the resource
opal.resource_view_reconnect(o, "CNSIM", "CNSIM4")
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.session *Create the R session*

Description

Create the R session if it does not exist and returns its identifier.

Usage

```
opal.session(opal, wait = TRUE)
```

Arguments

opal	Opal object.
wait	Wait for R session to be operational (default is TRUE)

Value

The R session identifier.

See Also

Other session functions: opal.session_delete(), opal.session_events(), opal.session_exists(), opal.session_get(), opal.session_running()

Examples

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

## End(Not run)
```

---

opal.session_delete *Delete the R session*

Description

Delete the remote R session. Ignored if the session does not exist.

Usage

```
opal.session_delete(opal)
```

Arguments

opal	Opal object.

See Also

Other session functions: opal.session(), opal.session_events(), opal.session_exists(), opal.session_get(), opal.session_running()

Examples

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

## End(Not run)
```

opal.session_events *Get the events associated to the R session*

Description

The remote R session logs events in its process of creation. Each event has a the current state of the R session, a date and a message reported by the R server. Fails if the session does not exist.

Usage

```
opal.session_events(opal)
```

Arguments

opal	Opal object.

Value

A data frame.

See Also

Other session functions: opal.session(), opal.session_delete(), opal.session_exists(), opal.session_get(), opal.session_running()

Examples

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

## End(Not run)
```

opal.session_exists *Check if the R session exists*

Description

Check if the remote R session exists

Usage

```
opal.session_exists(opal)
```

Arguments

opal	Opal object.

Value

A logical value.

See Also

Other session functions: opal.session(), opal.session_delete(), opal.session_events(), opal.session_get(), opal.session_running()

Examples

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

## End(Not run)
```

---

opal.session_get *Get the R session*

Description

Get the R session details if it exists

Usage

```
opal.session_get(opal)
```

Arguments

opal	Opal object.

Value

The R session object.

See Also

Other session functions: opal.session(), opal.session_delete(), opal.session_events(), opal.session_exists(), opal.session_running()

Examples

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

## End(Not run)
```

`opal.session_running` *Check if the R session is running*

Description

Check if the remote R session is running and ready to receive R commands. Fails if the session does not exist.

Usage

```
opal.session_running(opal)
```

Arguments

`opal`	Opal object.

Value

A logical value.

See Also

Other session functions: `opal.session()`, `opal.session_delete()`, `opal.session_events()`, `opal.session_exists()`, `opal.session_get()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.session(o, wait = FALSE)
ready <- opal.session_running(o)
while(!ready) {
  Sys.sleep(1)
  ready <- opal.session_running(o)
  cat(".")
}
opal.session_get(o)
opal.logout(o)

## End(Not run)
```

opal.sql *Execute a SQL query on tables*

Description

The SQL query can apply to raw tables and/or views and require the permission to view the values of these tables. When all tables belong to a project, it is possible to simplify the SQL query by providing the project name parameter. Otherwise the fully qualified table names (‘<project>.<table>‘) must be specified in the FROM statements.

Usage

```
opal.sql(opal, query, project = NULL, id.name = "_id")
```

Arguments

opal	Opal connection object.
query	The SQL query statement.
project	Project name where the table(s) are located. If not provided, the SQL query must refer to the full table name(s) (use backquotes to escape, see examples).
id.name	The name of the column representing the entity identifiers. Default is '_id'.

Value

The lists of columns and rows, as a data.frame.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# with project context
opal.sql(o,
  'select avg(LAB_HDL) as HDL_AVG, GENDER
     from CNSIM1
     where LAB_HDL is not null
     group by GENDER',
  'CNSIM')
# without project context
opal.sql(o,
  'select avg(LAB_HDL) as HDL_AVG, GENDER
     from `CNSIM.CNSIM1`
     where LAB_HDL is not null
     group by GENDER')
opal.logout(o)

## End(Not run)
```

opal.sql_history *SQL query execution history*

Description

Getting the SQL execution is for being able to re-execute a previously submitted own SQL query (regular users) and for auditing users SQL activity (administrators only).

Usage

```
opal.sql_history(
  opal,
  project = NULL,
  offset = 0,
  limit = 100,
  user = NULL,
  df = TRUE
)
```

Arguments

opal	Opal connection object.
project	Project name used as the SQL execution context, to filter. If not specified, history from any context is returned. If NA is specified, the history of SQL executions without context is returned. Default is NULL.
offset	Number of history items to skip. Default is 0 (note that the items are ordered by most recent first).
limit	Maximum number of history items to return. Default is 100.
user	Filter by user name, only administrators can retrieve SQL history of other users. If NA is specified, the SQL execution history of all users will be retrieved. Default is the current user name.
df	Result is a data.frame or a list of raw data.

Value

A data frame.

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# with project context
opal.sql_history(o, 'CNSIM')
# without project context
opal.sql_history(o, NA)
# with or without project context
```

```
opal.sql_history(o)
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.datasource()`, `opal.datasources()`, `opal.tables()`, `opal.valueset()`, `opal.variable()`, `opal.variable_summary()`, `opal.variables()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table(o, 'CNSIM', '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.datasource(), opal.datasources(), opal.table(), opal.valueset(), opal.variable(), opal.variable_summary(), opal.variables()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.tables(o, 'CNSIM')
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 an Opal table or view*

Description

Create an 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 will be located.
table	Table name to be created
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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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,
  complete = FALSE
)
```

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.
`complete`	A logical to indicate if the provided dictionary is complete, i.e. that the variables not included in this dictionary must be removed from the view. Otherwise (default) only the addition or the update of variables is performed.

See Also

Other table functions: `opal.perms()`, `opal.resource_view_create()`, `opal.resource_view_reconnect()`, `opal.table_create()`, `opal.table_delete()`, `opal.table_dictionary_get()`, `opal.table_exists()`, `opal.table_export()`, `opal.table_get()`, `opal.table_import()`, `opal.table_perm()`, `opal.table_perm_add()`, `opal.table_perm_delete()`, `opal.table_save()`, `opal.table_truncate()`, `opal.table_view_create()`, `opal.table_view_update()`

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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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_export *Export a table as a file*

Description

Export a table as file in the specified format. The file destination is in the Opal server file system. See opal.file_download to download the file locally. See also opal.table_get to get directly the table as an R object.

Usage

```
opal.table_export(
  opal,
  project,
  table,
  file,
  identifiers = NULL,
  id.name = "id",
  wait = TRUE
)
```

Arguments

opal	Opal connection object.
project	Project name where the table is located.
table	Table name to export.
file	Destination file in the Opal file system. The expected file extensions are: rds (RDS), sav (SPSS), zsav (SPSS compressed), sas7bdat (SAS), xpt (SAS Transport), dta (Stata).RDS (serialized single R object) is to be read by base::readRDS(), while other formats are supported by the haven R package.
identifiers	Name of the identifiers mapping to use when exporting entities from Opal.
id.name	The name of the column representing the entity identifiers. Default is 'id'.
wait	Wait for import task completion. Default is TRUE.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
cqx <- opal.table_export(o, "CNSIM", "CNSIM1",
                         file = "/home/administrator/cnsim1.sav")
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,
  id.name = "id",
  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.
id.name	The name of the column representing the entity identifiers. Default is 'id'. Requires Opal 4.0+.
variables	(Deprecated) List of variable names or Javascript expression that selects the variables of a table (ignored if value does not refer to a table). See javascript documentation: http://wiki.obiba.org/display/OPALDOC/Variable+Methods
missings	(Deprecated) Include the missing values (default is TRUE).

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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_import *Import a file as table*

Description

Import a file as a table in Opal. The file formats supported are: RDS (.rds), SPSS (.sav), SPSS compressed (.zsav), SAS (.sas7bdat), SAS Transport (.xpt), Stata (.dta). The RDS format is a serialized single R object (expected to be of tibble class), that can be obtained using base::saveRDS(). The other file formats are the ones supported by the haven R package. This operation creates an importation task in Opal that can be followed (see tasks related functions).

Usage

```
opal.table_import(
  opal,
  file,
  project,
  table,
  identifiers = NULL,
  policy = "required",
  id.name = "id",
  type = "Participant",
  wait = TRUE
)
```

Arguments

opal	Opal object.
file	Path in Opal to the file that will be read as a tibble.
project	Name of the project into which the data are to be imported.

table	Destination table name.
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 table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.table_import(o, '/home/administrator/mydataset.rds', 'test', 'mytable')
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

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_save(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_truncate(), opal.table_view_create(), opal.table_view_update()

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.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_view_create(), opal.table_view_update()

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.table_view_create` *Create an Opal view over tables*

Description

Create an Opal view if a table with same name does not already exist. The view created will have no dictionary, use opal.table_dictionary_update to apply a dictionary.

Usage

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

Arguments

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

See Also

Other table functions: `opal.perms()`, `opal.resource_view_create()`, `opal.resource_view_reconnect()`, `opal.table_create()`, `opal.table_delete()`, `opal.table_dictionary_get()`, `opal.table_dictionary_update()`, `opal.table_exists()`, `opal.table_export()`, `opal.table_get()`, `opal.table_import()`, `opal.table_perm()`, `opal.table_perm_add()`, `opal.table_perm_delete()`, `opal.table_save()`, `opal.table_truncate()`, `opal.table_view_update()`

Examples

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

## End(Not run)
```

opal.table_view_update *Update the table references and the entity filter of an Opal view*

Description

Update the table references and/or the entity filter of an existing Opal view. The view dictionary will NOT be modified (use opal.table_dictionary_update to apply a dictionary).

Usage

```
opal.table_view_update(opal, project, table, tables = NULL, where = NULL)
```

Arguments

opal	Opal connection object.
project	Project name where the table will be located.
table	Table name to be created.
tables	List of the fully qualified table names that are referred by the view. Not modified when NULL (default).
where	The entity filter script. Not modified when NULL (default). To remove the filter, set an empty string.

See Also

Other table functions: opal.perms(), opal.resource_view_create(), opal.resource_view_reconnect(), opal.table_create(), opal.table_delete(), opal.table_dictionary_get(), opal.table_dictionary_update(), opal.table_exists(), opal.table_export(), opal.table_get(), opal.table_import(), opal.table_perm(), opal.table_perm_add(), opal.table_perm_delete(), opal.table_save(), opal.table_truncate(), opal.table_view_create()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# make a view
opal.table_view_create(o, "CNSIM", "CNSIM123",
                       c("CNSIM.CNSIM1"))

# update the table references
opal.table_view_update(o, "CNSIM", "CNSIM123",
                       tables = c("CNSIM.CNSIM1", "CNSIM.CNSIM2", "CNSIM.CNSIM3"))

# update the entity filter
opal.table_view_update(o, "CNSIM", "CNSIM123", where = "$('LAB_TSC').ge(5)")

# remove the entity filter
```

```
opal.table_view_update(o, "CNSIM", "CNSIM123", where = "")

# update both the table references and the entity filter
opal.table_view_update(o, "CNSIM", "CNSIM123",
                       tables = c("CNSIM.CNSIM1", "CNSIM.CNSIM2", "CNSIM.CNSIM3"),
                       where = "$('LAB_TSC').ge(5)")
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_delete()`, `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(), opal.task_cancel(), opal.task_delete(), opal.task_wait()

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(), opal.task_delete(), opal.task_wait(), opal.tasks()

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_delete *Delete a task*

Description

Delete a specific task.

Usage

```
opal.task_delete(opal, id)
```

Arguments

opal	Opal object.
id	Task identifier.

See Also

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

Examples

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

## End(Not run)
```

opal.task_wait *Wait for a task to complete or anticipate its cancellation.*

Description

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

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(), opal.task_cancel(), opal.task_delete(), opal.tasks()

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(), 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.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(), 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_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(), opal.taxonomy_delete(), 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_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(), opal.taxonomy_delete(), opal.taxonomy_download(), 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()`, `opal.taxonomy_delete()`, `opal.taxonomy_download()`, `opal.taxonomy_upload()`, `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.token *Get a personal access token*

Description

Get a personal access token details. Like for the other token functions, this operation requires the user to authenticate with username/password credentials.

Usage

```
opal.token(opal, name)
```

Arguments

opal	Opal object.
name	Name of the token

See Also

Other token functions: opal.token_datashield_create(), opal.token_delete(), opal.token_r_create(), opal.token_renew(), opal.token_sql_create(), opal.tokens()

Examples

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

## End(Not run)
```

opal.tokens *Get the personal access tokens*

Description

Get the list of personal access tokens. Like for the other token functions, this operation requires the user to authenticate with username/password credentials.

Usage

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

Arguments

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

See Also

Other token functions: opal.token(), opal.token_datashield_create(), opal.token_delete(), opal.token_r_create(), opal.token_renew(), opal.token_sql_create()

Examples

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

## End(Not run)
```

---

opal.token_datashield_create

Create a personal access token for Datashield usage

Description

Create a personal access token for Datashield usage. Like for the other token functions, this operation requires the user to authenticate with username/password credentials.

Usage

```
opal.token_datashield_create(opal, name, projects = NULL)
```

Arguments

opal	Opal object.
name	Name of the token
projects	Vector of project names, to which the token applies. Default is NULL (all projects).

Value

The token value.

See Also

Other token functions: opal.token(), opal.token_delete(), opal.token_r_create(), opal.token_renew(), opal.token_sql_create(), opal.tokens()

Examples

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

## End(Not run)
```

`opal.token_delete` *Delete a personal access token*

Description

Delete a personal access token permanently. Like for the other token functions, this operation requires the user to authenticate with username/password credentials.

Usage

```
opal.token_delete(opal, name)
```

Arguments

`opal`	Opal object.
`name`	Name of the token

See Also

Other token functions: `opal.token()`, `opal.token_datashield_create()`, `opal.token_r_create()`, `opal.token_renew()`, `opal.token_sql_create()`, `opal.tokens()`

Examples

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

## End(Not run)
```

opal.token_renew *Renew an inactive personal access token*

Description

Renew an inactive personal access token after it has been marked as being inactive. Like for the other token functions, this operation requires the user to authenticate with username/password credentials.

Usage

```
opal.token_renew(opal, name)
```

Arguments

opal	Opal object.
name	Name of the token

See Also

Other token functions: opal.token(), opal.token_datashield_create(), opal.token_delete(), opal.token_r_create(), opal.token_sql_create(), opal.tokens()

Examples

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

## End(Not run)
```

opal.token_r_create *Create a personal access token for R usage*

Description

Create a personal access token for R (server) usage. Like for the other token functions, this operation requires the user to authenticate with username/password credentials.

Usage

```
opal.token_r_create(
  opal,
  name,
  projects = NULL,
  access = NULL,
  commands = c("export")
)
```

Arguments

opal	Opal object.
name	Name of the token
projects	Vector of project names, to which the token applies. Default is NULL (all projects).
access	Data access level: 'READ' (read-only) or 'READ_NO_VALUES' (read-only, without access to individual-level data) or NULL (default).
commands	Task commands that can launched on a project: 'import' and/or 'export'. Default is 'export' (use NULL for no task commands).

Value

The token value.

See Also

Other token functions: opal.token(), opal.token_datashield_create(), opal.token_delete(), opal.token_renew(), opal.token_sql_create(), opal.tokens()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
token <- opal.token_r_create(o, 'r-1', access = 'READ', commands = 'export')
opal.logout(o)

## End(Not run)
```

---

opal.token_sql_create *Create a personal access token for SQL usage*

Description

Create a personal access token for SQL usage. Like for the other token functions, this operation requires the user to authenticate with username/password credentials.

Usage

```
opal.token_sql_create(opal, name, projects = NULL)
```

Arguments

opal	Opal object.
name	Name of the token
projects	Vector of project names, to which the token applies. Default is NULL (all projects).

Value

The token value.

See Also

Other token functions: opal.token(), opal.token_datashield_create(), opal.token_delete(), opal.token_r_create(), opal.token_renew(), opal.tokens()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
token <- opal.token_sql_create(o, 'sql-1')
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(), opal.execute.source(), 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.datasource()`, `opal.datasources()`, `opal.table()`, `opal.tables()`, `opal.variable()`, `opal.variable_summary()`, `opal.variables()`

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.valueset(o, 'CNSIM', '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.datasource(), opal.datasources(), opal.table(), opal.tables(), opal.valueset(), opal.variable_summary(), opal.variables()

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.variable(o, 'CNSIM', '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.datasource(), opal.datasources(), opal.table(), opal.tables(), opal.valueset(), opal.variable(), opal.variable_summary()

Examples

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

## End(Not run)
```

---

opal.variable_summary *Get summary statistics of a variable of a table*

Description

Get summary statistics of a variable of a table

Usage

```
opal.variable_summary(
  opal,
  datasource,
  table,
  variable,
  cached = TRUE,
  nature = NULL
)
```

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.
cached	Get cached summary if exists. When FALSE, the cached summary is evicted and replaced by the newly calculated one. Default is TRUE.
nature	Force summary nature, independently from the variable. Possible values are: CATEGORICAL, CONTINUOUS, TEMPORAL, GEO, BINARY, UNDETERMINED.

See Also

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

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.variable_summary(o, 'CNSIM', 'CNSIM1', 'GENDER')
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()`, `opal.taxonomy_delete()`, `opal.taxonomy_download()`, `opal.taxonomy_upload()`, `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(), opal.taxonomy_delete(), opal.taxonomy_download(), opal.taxonomy_upload(), 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_restore(), 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_restore

Restore a R workspace from a opal.

Description

Restore a R workspace from a opal.

Usage

```
opal.workspace_restore(opal, ws)
```

Arguments

opal	Opal object.
ws	The workspace name

See Also

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

Examples

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

Value

The workspace ID (invisible)

See Also

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

Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
# provide a workspace ID
opal.workspace_save(o, 'test')
# or use default one
id <- opal.workspace_save(o)
opal.logout(o)

## End(Not run)
```

Index

∗ **DataSHIELD functions**
dsadmin.get_method, 13
dsadmin.get_methods, 14
dsadmin.get_options, 14
dsadmin.install_github_package, 16
dsadmin.install_local_package, 17
dsadmin.install_package, 18
dsadmin.installed_package, 15
dsadmin.package_description, 21
dsadmin.package_descriptions, 22
dsadmin.publish_package, 35
dsadmin.remove_package, 43
dsadmin.rm_method, 44
dsadmin.rm_methods, 45
dsadmin.rm_option, 45
dsadmin.rm_options, 46
dsadmin.rm_package_methods, 47
dsadmin.set_method, 48
dsadmin.set_option, 49
dsadmin.set_package_methods, 50
dsadmin.unpublish_package, 51
∗ **DataSHIELD profiles**
dsadmin.profile, 25
dsadmin.profile_access, 26
dsadmin.profile_create, 27
dsadmin.profile_delete, 28
dsadmin.profile_enable, 29
dsadmin.profile_exists, 29
dsadmin.profile_init, 30
dsadmin.profile_perm, 31
dsadmin.profile_perm_add, 32
dsadmin.profile_perm_delete, 33
dsadmin.profile_rparser, 34
dsadmin.profiles, 25
∗ **DataSHIELD quotas**
dsadmin.quota, 36
dsadmin.quota_delete, 38
dsadmin.quota_enable, 39
dsadmin.quota_exists, 40
dsadmin.quota_save, 41
dsadmin.quota_usage, 42
dsadmin.quotas, 37
opal.quotas, 122
∗ **REST functions**
opal.delete, 92
opal.get, 105
opal.post, 110
opal.put, 121
∗ **administration functions**
oadmin.install_bioconductor_package, 56
oadmin.install_cran_package, 57
oadmin.install_devtools, 57
oadmin.install_github_package, 58
oadmin.install_local_package, 59
oadmin.install_package, 60
oadmin.installed_devtools, 54
oadmin.installed_package, 54
oadmin.installed_packages, 55
oadmin.package_description, 63
oadmin.remove_package, 66
∗ **assignment functions**
opal.assign, 79
opal.assign.data, 80
opal.assign.resource, 81
opal.assign.script, 82
opal.assign.table, 82
opal.assign.table.tibble, 84
∗ **command functions**
opal.command, 87
opal.command_result, 89
opal.command_rm, 90
opal.commands, 88
opal.commands_rm, 88
∗ **connection functions**
opal.login, 106
opal.logout, 108
opal.profiles, 111

∗ **datasource functions**
opal.annotate, 77
opal.annotations, 78
opal.attribute_values, 86
opal.datasource, 90
opal.datasources, 91
opal.table, 148
opal.tables, 149
opal.valueset, 180
opal.variable, 181
opal.variable_summary, 182
opal.variables, 181
∗ **execution functions**
opal.execute, 92
opal.execute.source, 93
opal.load_package, 106
opal.unload_package, 179
∗ **file functions**
opal.file, 94
opal.file_cp, 95
opal.file_download, 95
opal.file_info, 96
opal.file_ls, 97
opal.file_mkdir, 98
opal.file_mkdir_tmp, 99
opal.file_mv, 99
opal.file_read, 100
opal.file_rm, 101
opal.file_unzip, 102
opal.file_upload, 103
opal.file_write, 104
∗ **project functions**
opal.project, 112
opal.project_create, 115
opal.project_delete, 117
opal.project_exists, 117
opal.projects, 112
opal.projects_databases, 113
∗ **resource functions**
opal.resource, 124
opal.resource_create, 128
opal.resource_delete, 129
opal.resource_exists, 130
opal.resource_extension_create, 131
opal.resource_get, 132
opal.resource_perm, 133
opal.resource_perm_add, 134
opal.resource_perm_delete, 135
opal.resources, 125
opal.resources_perm, 125
opal.resources_perm_add, 126
opal.resources_perm_delete, 127
∗ **session functions**
opal.session, 138
opal.session_delete, 139
opal.session_events, 140
opal.session_exists, 140
opal.session_get, 141
opal.session_running, 142
∗ **symbol functions**
opal.rm, 138
opal.symbol_import, 145
opal.symbol_rm, 146
opal.symbol_save, 147
opal.symbols, 145
∗ **table functions**
opal.perms, 109
opal.resource_view_create, 136
opal.resource_view_reconnect, 137
opal.table_create, 151
opal.table_delete, 152
opal.table_dictionary_get, 153
opal.table_dictionary_update, 154
opal.table_exists, 155
opal.table_export, 156
opal.table_get, 157
opal.table_import, 158
opal.table_perm, 159
opal.table_perm_add, 160
opal.table_perm_delete, 161
opal.table_save, 162
opal.table_truncate, 163
opal.table_view_create, 164
opal.table_view_update, 165
∗ **task functions**
opal.task, 166
opal.task_cancel, 167
opal.task_delete, 168
opal.task_wait, 169
opal.tasks, 167
∗ **taxonomy functions**
opal.taxonomies, 169
opal.taxonomy, 170
opal.taxonomy_delete, 171
opal.taxonomy_download, 171

opal.taxonomy_upload, 172
opal.terms, 173
opal.vocabularies, 184
opal.vocabulary, 185
∗ **token functions**
opal.token, 174
opal.token_datashield_create, 175
opal.token_delete, 176
opal.token_r_create, 177
opal.token_renew, 177
opal.token_sql_create, 178
opal.tokens, 174
∗ **user functions**
oadmin.user_add, 72
oadmin.user_delete, 72
oadmin.user_enable, 73
oadmin.user_exists, 74
oadmin.user_profile_delete, 75
oadmin.user_profiles, 75
oadmin.user_reset_password, 76
oadmin.users, 71
∗ **workspace functions**
opal.workspace_restore, 186
opal.workspace_rm, 187
opal.workspace_save, 187
opal.workspaces, 185

dictionary.annotate, 6
dictionary.annotate.harmo_status, 7
dictionary.annotations, 8
dictionary.apply, 9
dictionary.inspect, 10
dsadmin.activity, 11
dsadmin.activity_summary, 12
dsadmin.get_method, 13
dsadmin.get_method(), *14–17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.get_methods, 14
dsadmin.get_methods(), *13*, *15–17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.get_options, 14
dsadmin.get_options(), *13, 14*, *16, 17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.install_github_package, 16
dsadmin.install_github_package(), *13–17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.install_local_package, 17
dsadmin.install_local_package(), *13–17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.install_package, 18
dsadmin.install_package(), *13–17*, *21, 22*, *35*, *43–51*
dsadmin.installed_package, 15
dsadmin.installed_package(), *13–15*, *17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.log, 19
dsadmin.package_description, 21
dsadmin.package_description(), *13–17*, *19*, *22*, *35*, *43–51*
dsadmin.package_descriptions, 22
dsadmin.package_descriptions(), *13–17*, *19*, *21*, *35*, *43–51*
dsadmin.perm, 23
dsadmin.perm_add, 23
dsadmin.perm_delete, 24
dsadmin.profile, 25
dsadmin.profile(), *26–34*
dsadmin.profile_access, 26
dsadmin.profile_access(), *25–34*
dsadmin.profile_create, 27
dsadmin.profile_create(), *25–34*
dsadmin.profile_delete, 28
dsadmin.profile_delete(), *25–27*, *29–34*
dsadmin.profile_enable, 29
dsadmin.profile_enable(), *25–28*, *30–34*
dsadmin.profile_exists, 29
dsadmin.profile_exists(), *25–34*
dsadmin.profile_init, 30
dsadmin.profile_init(), *25–34*
dsadmin.profile_perm, *26*, 31
dsadmin.profile_perm(), *25–30*, *32–34*
dsadmin.profile_perm_add, 32
dsadmin.profile_perm_add(), *25–31*, *33, 34*
dsadmin.profile_perm_delete, 33
dsadmin.profile_perm_delete(), *25–32*, *34*
dsadmin.profile_rparser, 34
dsadmin.profile_rparser(), *25–33*
dsadmin.profiles, *13–18*, *21, 22*, 25, *35*, *43–51*
dsadmin.profiles(), *25*, *27–34*
dsadmin.publish_package, *30*, 35
dsadmin.publish_package(), *13–17*, *19*, *21, 22*, *43–51*
dsadmin.quota, 36
dsadmin.quota(), *37–40*, *42, 43*, *123*

dsadmin.quota_delete, 38
dsadmin.quota_delete(), *36, 37*, *39, 40*, *42, 43*, *123*
dsadmin.quota_enable, 39
dsadmin.quota_enable(), *36–38*, *40*, *42, 43*, *123*
dsadmin.quota_exists, 40
dsadmin.quota_exists(), *36–39*, *42, 43*, *123*
dsadmin.quota_save, 41
dsadmin.quota_save(), *36–40*, *43*, *123*
dsadmin.quota_usage, *36*, 42, *123*
dsadmin.quota_usage(), *36–40*, *42*, *123*
dsadmin.quotas, 37
dsadmin.quotas(), *36*, *38–40*, *42, 43*, *123*
dsadmin.remove_package, 43
dsadmin.remove_package(), *13–17*, *19*, *21, 22*, *35*, *44–51*
dsadmin.rm_method, 44
dsadmin.rm_method(), *13–17*, *19*, *21, 22*, *35*, *43*, *45–51*
dsadmin.rm_methods, 45
dsadmin.rm_methods(), *13–17*, *19*, *21, 22*, *35*, *43, 44*, *46–51*
dsadmin.rm_option, 45
dsadmin.rm_option(), *13–17*, *19*, *21, 22*, *35*, *43–45*, *47–51*
dsadmin.rm_options, 46
dsadmin.rm_options(), *13–17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.rm_package_methods, 47
dsadmin.rm_package_methods(), *13–17*, *19*, *21, 22*, *35*, *43–51*
dsadmin.set_method, 48
dsadmin.set_method(), *13–17*, *19*, *21, 22*, *35*, *43–47*, *49–51*
dsadmin.set_option, *30*, 49
dsadmin.set_option(), *13–17*, *19*, *21, 22*, *35*, *43–48*, *50, 51*
dsadmin.set_package_methods, *30*, 50
dsadmin.set_package_methods(), *13–17*, *19*, *21, 22*, *35*, *43–49*, *51*
dsadmin.unpublish_package, 51
dsadmin.unpublish_package(), *13–17*, *19*, *21, 22*, *35*, *43–50*

oadmin.activity, 52
oadmin.activity_summary, 53
oadmin.install_bioconductor_package, 56
oadmin.install_bioconductor_package(), *54–60*, *63*, *66*
oadmin.install_cran_package, 57
oadmin.install_cran_package(), *54–56*, *58–60*, *63*, *66*
oadmin.install_devtools, 57
oadmin.install_devtools(), *54–57*, *59, 60*, *63*, *66*
oadmin.install_github_package, 58
oadmin.install_github_package(), *54–60*, *63*, *66*
oadmin.install_local_package, 59
oadmin.install_local_package(), *54–60*, *63*, *66*
oadmin.install_package, 60
oadmin.install_package(), *54–59*, *63*, *66*
oadmin.installed_devtools, 54
oadmin.installed_devtools(), *55–60*, *63*, *66*
oadmin.installed_package, 54
oadmin.installed_package(), *54*, *56–60*, *63*, *66*
oadmin.installed_packages, 55
oadmin.installed_packages(), *54–60*, *63*, *66*
oadmin.log, 61
oadmin.log_rest, 61
oadmin.log_sql, 62
oadmin.package_description, 63
oadmin.package_description(), *54–60*, *66*
oadmin.perm, 64
oadmin.perm_add, 64
oadmin.perm_delete, 65
oadmin.r_perm, *64*, 66
oadmin.r_perm_add, *64*, 67
oadmin.r_perm_delete, *65*, 68
oadmin.remove_package, 66
oadmin.remove_package(), *54–60*, *63*
oadmin.system_metrics, 68
oadmin.system_perm, 69
oadmin.system_perm_add, 70
oadmin.system_perm_delete, 70
oadmin.user_add, 72
oadmin.user_add(), *71*, *73–76*
oadmin.user_delete, 72
oadmin.user_delete(), *71–76*

oadmin.user_enable, 73
oadmin.user_enable(), *71–76*
oadmin.user_exists, 74
oadmin.user_exists(), *71–73*, *75, 76*
oadmin.user_profile_delete, 75
oadmin.user_profile_delete(), *71–76*
oadmin.user_profiles, *74*, 75
oadmin.user_profiles(), *71–74*, *76*
oadmin.user_reset_password, 76
oadmin.user_reset_password(), *71–76*
oadmin.users, 71, *74*
oadmin.users(), *72–76*
opal.annotate, 77
opal.annotate(), *78*, *86*, *91*, *148, 149*, *180–183*
opal.annotations, *77*, 78
opal.annotations(), *77*, *86*, *91*, *148, 149*, *180–183*
opal.as_md_table, 85
opal.assign, 79
opal.assign(), *80–83*, *85*
opal.assign.data, 80
opal.assign.data(), *79*, *81–83*, *85*
opal.assign.resource, 81
opal.assign.resource(), *79, 80*, *82, 83*, *85*
opal.assign.script, 82
opal.assign.script(), *79–81*, *83*, *85*
opal.assign.table, 82
opal.assign.table(), *79–82*, *85*
opal.assign.table.tibble, 84
opal.assign.table.tibble(), *79–83*
opal.attribute_values, 86
opal.attribute_values(), *77, 78*, *91*, *148, 149*, *180–183*
opal.command, 87
opal.command(), *88–90*
opal.command_result, 89
opal.command_result(), *87, 88*, *90*
opal.command_rm, 90
opal.command_rm(), *87–89*
opal.commands, 88
opal.commands(), *87–90*
opal.commands_rm, 88
opal.commands_rm(), *87–90*
opal.datasource, 90
opal.datasource(), *77, 78*, *86*, *91*, *148, 149*, *180–183*
opal.datasources, 91
opal.datasources(), *77, 78*, *86*, *91*, *148, 149*, *180–183*
opal.delete, 92
opal.delete(), *105*, *110*, *122*
opal.execute, 92
opal.execute(), *93*, *106*, *179*
opal.execute.source, 93
opal.execute.source(), *93*, *106*, *179*
opal.file, 94
opal.file(), *95–104*
opal.file_cp, 95
opal.file_cp(), *94*, *96–104*
opal.file_download, 95, *156*
opal.file_download(), *94, 95*, *97–104*
opal.file_info, 96
opal.file_info(), *94–104*
opal.file_ls, 97
opal.file_ls(), *94–104*
opal.file_mkdir, 98
opal.file_mkdir(), *94–97*, *99–104*
opal.file_mkdir_tmp, 99
opal.file_mkdir_tmp(), *94–98*, *100–104*
opal.file_mv, 99
opal.file_mv(), *94–99*, *101–104*
opal.file_read, 100
opal.file_read(), *94–104*
opal.file_rm, 101
opal.file_rm(), *94–104*
opal.file_unzip, 102
opal.file_unzip(), *94–101*, *103, 104*
opal.file_upload, 103
opal.file_upload(), *94–102*, *104*
opal.file_write, 104
opal.file_write(), *94–103*
opal.get, 105
opal.get(), *92*, *110*, *122*
opal.load_package, 106
opal.load_package(), *93*, *179*
opal.login, 106, *111*
opal.login(), *109*, *111*
opal.logout, 108
opal.logout(), *107*, *111*
opal.perms, 109
opal.perms(), *136, 137*, *152–155*, *157–161*, *163–165*
opal.post, 110
opal.post(), *92*, *105*, *122*
opal.profiles, *54–60*, *63*, *66*, *107*, 111

opal.profiles(), *107*, *109*
opal.project, 112
opal.project(), *113*, *116–118*
opal.project_backup, 114
opal.project_command, *114*, 115, *121*
opal.project_create, 115
opal.project_create(), *112*, *113*, *117*, *118*
opal.project_delete, 117
opal.project_delete(), *112*, *113*, *116*, *118*
opal.project_exists, 117
opal.project_exists(), *112*, *113*, *116*, *117*
opal.project_perm, 118
opal.project_perm_add, 119
opal.project_perm_delete, 120
opal.project_restore, 120
opal.projects, 112
opal.projects(), *112*, *113*, *116–118*
opal.projects_databases, 113
opal.projects_databases(), *112*, *113*, *116–118*
opal.put, 121
opal.put(), *92*, *105*, *110*
opal.quotas, *42*, 122
opal.quotas(), *36–40*, *42*, *43*
opal.report, 123
opal.resource, 124
opal.resource(), *125–127*, *129–135*
opal.resource_create, 128
opal.resource_create(), *124–127*, *129–135*
opal.resource_delete, 129
opal.resource_delete(), *124–127*, *129–135*
opal.resource_exists, 130
opal.resource_exists(), *124–127*, *129*, *131–135*
opal.resource_extension_create, 131
opal.resource_extension_create(), *124–127*, *129*, *130*, *132–135*
opal.resource_get, 132
opal.resource_get(), *124–127*, *129–131*, *133–135*
opal.resource_perm, 133
opal.resource_perm(), *124–127*, *129–132*, *134*, *135*
opal.resource_perm_add, 134
opal.resource_perm_add(), *124–127*, *129–133*, *135*
opal.resource_perm_delete, 135
opal.resource_perm_delete(), *124–127*, *129–134*
opal.resource_view_create, 136
opal.resource_view_create(), *109*, *137*, *152–155*, *157–161*, *163–165*
opal.resource_view_reconnect, 137
opal.resource_view_reconnect(), *109*, *136*, *152–155*, *157–161*, *163–165*
opal.resources, 125
opal.resources(), *124*, *126*, *127*, *129–135*
opal.resources_perm, 125
opal.resources_perm(), *124*, *125*, *127*, *129–135*
opal.resources_perm_add, 126
opal.resources_perm_add(), *124–127*, *129–135*
opal.resources_perm_delete, 127
opal.resources_perm_delete(), *124–127*, *129–135*
opal.rm, 138
opal.rm(), *145–147*
opal.session, 138
opal.session(), *139–142*
opal.session_delete, 139
opal.session_delete(), *139–142*
opal.session_events, 140
opal.session_events(), *139*, *141*, *142*
opal.session_exists, 140
opal.session_exists(), *139–142*
opal.session_get, 141
opal.session_get(), *139–142*
opal.session_running, 142
opal.session_running(), *139–141*
opal.sql, 143
opal.sql_history, 144
opal.symbol_import, 145
opal.symbol_import(), *138*, *145*, *147*
opal.symbol_rm, 146
opal.symbol_rm(), *138*, *145–147*
opal.symbol_save, 147
opal.symbol_save(), *138*, *145–147*
opal.symbols, 145
opal.symbols(), *138*, *146*, *147*
opal.table, 148
opal.table(), *77*, *78*, *86*, *91*, *149*, *180–183*
opal.table_create, 151
opal.table_create(), *109*, *136*, *137*,

153–155, *157–161*, *163–165*
opal.table_delete, 152, *163*
opal.table_delete(), *109*, *136, 137*, *152–155*, *157–161*, *163–165*
opal.table_dictionary_get, 153
opal.table_dictionary_get(), *109*, *136, 137*, *152–155*, *157–161*, *163–165*
opal.table_dictionary_update, *136*, *151*, 154, *164, 165*
opal.table_dictionary_update(), *109*, *136, 137*, *152, 153*, *155*, *157–161*, *163–165*
opal.table_exists, 155
opal.table_exists(), *109*, *136, 137*, *152–154*, *157–161*, *163–165*
opal.table_export, 156
opal.table_export(), *109*, *136, 137*, *152–155*, *158–161*, *163–165*
opal.table_get, *156*, 157
opal.table_get(), *109*, *136, 137*, *152–155*, *157*, *159–161*, *163–165*
opal.table_import, 158
opal.table_import(), *109*, *136, 137*, *152–155*, *157, 158*, *160, 161*, *163–165*
opal.table_perm, 159
opal.table_perm(), *109*, *136, 137*, *152–155*, *157–161*, *163–165*
opal.table_perm_add, 160
opal.table_perm_add(), *109*, *136, 137*, *152–155*, *157–161*, *163–165*
opal.table_perm_delete, 161
opal.table_perm_delete(), *109*, *136, 137*, *152–155*, *157–160*, *163–165*
opal.table_save, 162
opal.table_save(), *109*, *136, 137*, *152–155*, *157–161*, *163–165*
opal.table_truncate, *152*, *162*, 163
opal.table_truncate(), *109*, *136, 137*, *152–155*, *157–161*, *163–165*
opal.table_view_create, 164
opal.table_view_create(), *109*, *136, 137*, *152–155*, *157–161*, *163*, *165*
opal.table_view_update, 165
opal.table_view_update(), *109*, *136, 137*, *152–155*, *157–161*, *163, 164*
opal.tables, 149
opal.tables(), *77, 78*, *86*, *91*, *148*, *180–183*
opal.tables_perm, 149
opal.tables_perm_add, 150
opal.tables_perm_delete, 151
opal.task, 166
opal.task(), *167–169*
opal.task_cancel, 167
opal.task_cancel(), *166–169*
opal.task_delete, 168
opal.task_delete(), *166–169*
opal.task_wait, 169
opal.task_wait(), *166–168*
opal.tasks, 167
opal.tasks(), *166*, *168, 169*
opal.taxonomies, 169
opal.taxonomies(), *170–173*, *184, 185*
opal.taxonomy, 170
opal.taxonomy(), *170–173*, *184, 185*
opal.taxonomy_delete, 171, *172*
opal.taxonomy_delete(), *170*, *172, 173*, *184, 185*
opal.taxonomy_download, 171
opal.taxonomy_download(), *170–173*, *184*, *185*
opal.taxonomy_upload, 172
opal.taxonomy_upload(), *170–173*, *184*, *185*
opal.terms, 173
opal.terms(), *170–172*, *184, 185*
opal.token, 174
opal.token(), *175–179*
opal.token_datashield_create, 175
opal.token_datashield_create(), *174–179*
opal.token_delete, 176
opal.token_delete(), *174, 175*, *177–179*
opal.token_r_create, 177
opal.token_r_create(), *174–177*, *179*
opal.token_renew, 177
opal.token_renew(), *174–176*, *178, 179*
opal.token_sql_create, 178
opal.token_sql_create(), *174–178*
opal.tokens, 174
opal.tokens(), *174–179*
opal.unload_package, 179
opal.unload_package(), *93*, *106*
opal.valueset, 180
opal.valueset(), *77, 78*, *86*, *91*, *148, 149*, *181–183*

opal.variable, 181
opal.variable(), *77, 78*, *86*, *91*, *148, 149*, *180*, *182, 183*
opal.variable_summary, 182
opal.variable_summary(), *77, 78*, *86*, *91*, *148, 149*, *180–182*
opal.variables, 181
opal.variables(), *77, 78*, *86*, *91*, *148, 149*, *180, 181*, *183*
opal.version_compare, 183
opal.vocabularies, 184
opal.vocabularies(), *170–173*, *185*
opal.vocabulary, 185
opal.vocabulary(), *170–173*, *184*
opal.workspace_restore, 186
opal.workspace_restore(), *186–188*
opal.workspace_rm, 187
opal.workspace_rm(), *186*, *188*
opal.workspace_save, 187
opal.workspace_save(), *186, 187*
opal.workspaces, 185
opal.workspaces(), *186–188*