

# Package ‘opalr’

October 6, 2023

**Version** 3.4.0

**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://www.datashield.org/>

**BugReports** <https://github.com/obiba/opalr/issues/>

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**Encoding** UTF-8

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2023-10-06 14:20:02 UTC

## R topics documented:

|  |    |
|--|----|
| 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.remove_package . . . . .           | 36 |
| dsadmin.rm_method . . . . .                | 36 |
| dsadmin.rm_methods . . . . .               | 37 |
| dsadmin.rm_option . . . . .                | 38 |
| dsadmin.rm_options . . . . .               | 39 |
| dsadmin.rm_package_methods . . . . .       | 40 |
| dsadmin.set_method . . . . .               | 41 |
| dsadmin.set_option . . . . .               | 42 |

|   |    |
|---|----|
| dsadmin.set_package_methods . . . . .         | 43 |
| dsadmin.unpublish_package . . . . .           | 44 |
| oadmin.activity . . . . .                     | 45 |
| oadmin.activity_summary . . . . .             | 46 |
| oadmin.installed_devtools . . . . .           | 47 |
| oadmin.installed_package . . . . .            | 47 |
| oadmin.installed_packages . . . . .           | 48 |
| oadmin.install_bioconductor_package . . . . . | 49 |
| oadmin.install_cran_package . . . . .         | 50 |
| oadmin.install_devtools . . . . .             | 50 |
| oadmin.install_github_package . . . . .       | 51 |
| oadmin.install_local_package . . . . .        | 52 |
| oadmin.install_package . . . . .              | 53 |
| oadmin.log . . . . .                          | 54 |
| oadmin.log_rest . . . . .                     | 54 |
| oadmin.log_sql . . . . .                      | 55 |
| oadmin.package_description . . . . .          | 56 |
| oadmin.perm . . . . .                         | 57 |
| oadmin.perm_add . . . . .                     | 57 |
| oadmin.perm_delete . . . . .                  | 58 |
| oadmin.remove_package . . . . .               | 59 |
| oadmin.r_perm . . . . .                       | 59 |
| oadmin.r_perm_add . . . . .                   | 60 |
| oadmin.r_perm_delete . . . . .                | 61 |
| oadmin.system_metrics . . . . .               | 61 |
| oadmin.system_perm . . . . .                  | 62 |
| oadmin.system_perm_add . . . . .              | 63 |
| oadmin.system_perm_delete . . . . .           | 63 |
| oadmin.users . . . . .                        | 64 |
| oadmin.user_add . . . . .                     | 65 |
| oadmin.user_delete . . . . .                  | 65 |
| oadmin.user_enable . . . . .                  | 66 |
| oadmin.user_exists . . . . .                  | 67 |
| oadmin.user_profiles . . . . .                | 68 |
| oadmin.user_profile_delete . . . . .          | 68 |
| oadmin.user_reset_password . . . . .          | 69 |
| opal.annotate . . . . .                       | 70 |
| opal.annotations . . . . .                    | 71 |
| opal.assign . . . . .                         | 72 |
| opal.assign.data . . . . .                    | 73 |
| opal.assign.resource . . . . .                | 74 |
| opal.assign.script . . . . .                  | 75 |
| opal.assign.table . . . . .                   | 75 |
| opal.assign.table.tibble . . . . .            | 77 |
| opal.as_md_table . . . . .                    | 78 |
| opal.attribute_values . . . . .               | 79 |
| opal.command . . . . .                        | 80 |
| opal.commands . . . . .                       | 81 |

|                                      |     |
|--------------------------------------|-----|
| opal.commands_rm . . . . .           | 81  |
| opal.command_result . . . . .        | 82  |
| opal.command_rm . . . . .            | 83  |
| opal.datasource . . . . .            | 83  |
| opal.datasources . . . . .           | 84  |
| opal.delete . . . . .                | 85  |
| opal.execute . . . . .               | 85  |
| opal.execute.source . . . . .        | 86  |
| opal.file . . . . .                  | 87  |
| opal.file_cp . . . . .               | 88  |
| opal.file_download . . . . .         | 88  |
| opal.file_ls . . . . .               | 89  |
| opal.file_mkdir . . . . .            | 90  |
| opal.file_mkdir_tmp . . . . .        | 91  |
| opal.file_mv . . . . .               | 91  |
| opal.file_read . . . . .             | 92  |
| opal.file_rm . . . . .               | 93  |
| opal.file_unzip . . . . .            | 94  |
| opal.file_upload . . . . .           | 95  |
| opal.file_write . . . . .            | 96  |
| opal.get . . . . .                   | 97  |
| opal.load_package . . . . .          | 98  |
| opal.login . . . . .                 | 98  |
| opal.logout . . . . .                | 100 |
| opal.perms . . . . .                 | 101 |
| opal.post . . . . .                  | 102 |
| opal.profiles . . . . .              | 103 |
| opal.project . . . . .               | 103 |
| opal.projects . . . . .              | 104 |
| opal.projects_databases . . . . .    | 105 |
| opal.project_backup . . . . .        | 105 |
| opal.project_command . . . . .       | 106 |
| opal.project_create . . . . .        | 107 |
| opal.project_delete . . . . .        | 108 |
| opal.project_exists . . . . .        | 109 |
| opal.project_perm . . . . .          | 110 |
| opal.project_perm_add . . . . .      | 110 |
| opal.project_perm_delete . . . . .   | 111 |
| opal.project_restore . . . . .       | 112 |
| opal.put . . . . .                   | 113 |
| opal.report . . . . .                | 114 |
| opal.resource . . . . .              | 115 |
| opal.resources . . . . .             | 115 |
| opal.resources_perm . . . . .        | 116 |
| opal.resources_perm_add . . . . .    | 117 |
| opal.resources_perm_delete . . . . . | 118 |
| opal.resource_create . . . . .       | 119 |
| opal.resource_delete . . . . .       | 120 |

|  |     |
|--|-----|
| opal.resource_exists . . . . .           | 121 |
| opal.resource_extension_create . . . . . | 122 |
| opal.resource_get . . . . .              | 123 |
| opal.resource_perm . . . . .             | 124 |
| opal.resource_perm_add . . . . .         | 125 |
| opal.resource_perm_delete . . . . .      | 126 |
| opal.resource_view_create . . . . .      | 127 |
| opal.resource_view_reconnect . . . . .   | 128 |
| opal.rm . . . . .                        | 129 |
| opal.sql . . . . .                       | 129 |
| opal.sql_history . . . . .               | 130 |
| opal.symbols . . . . .                   | 131 |
| opal.symbol_import . . . . .             | 132 |
| opal.symbol_rm . . . . .                 | 133 |
| opal.symbol_save . . . . .               | 134 |
| opal.table . . . . .                     | 134 |
| opal.tables . . . . .                    | 135 |
| opal.tables_perm . . . . .               | 136 |
| opal.tables_perm_add . . . . .           | 137 |
| opal.tables_perm_delete . . . . .        | 137 |
| opal.table_create . . . . .              | 138 |
| opal.table_delete . . . . .              | 139 |
| opal.table_dictionary_get . . . . .      | 140 |
| opal.table_dictionary_update . . . . .   | 140 |
| opal.table_exists . . . . .              | 142 |
| opal.table_export . . . . .              | 143 |
| opal.table_get . . . . .                 | 144 |
| opal.table_import . . . . .              | 145 |
| opal.table_perm . . . . .                | 146 |
| opal.table_perm_add . . . . .            | 147 |
| opal.table_perm_delete . . . . .         | 148 |
| opal.table_save . . . . .                | 149 |
| opal.table_truncate . . . . .            | 150 |
| opal.table_view_create . . . . .         | 151 |
| opal.table_view_update . . . . .         | 152 |
| opal.task . . . . .                      | 153 |
| opal.tasks . . . . .                     | 154 |
| opal.task_cancel . . . . .               | 154 |
| opal.task_wait . . . . .                 | 155 |
| opal.taxonomies . . . . .                | 156 |
| opal.taxonomy . . . . .                  | 156 |
| opal.taxonomy_delete . . . . .           | 157 |
| opal.taxonomy_download . . . . .         | 158 |
| opal.taxonomy_upload . . . . .           | 158 |
| opal.terms . . . . .                     | 159 |
| opal.token . . . . .                     | 160 |
| opal.tokens . . . . .                    | 161 |
| opal.token_datashield_create . . . . .   | 161 |

|                                  |     |
|----------------------------------|-----|
| opal.token_delete . . . . .      | 162 |
| opal.token_renew . . . . .       | 163 |
| opal.token_r_create . . . . .    | 164 |
| opal.token_sql_create . . . . .  | 165 |
| opal.unload_package . . . . .    | 166 |
| opal.valueset . . . . .          | 166 |
| opal.variable . . . . .          | 167 |
| opal.variables . . . . .         | 168 |
| opal.variable_summary . . . . .  | 168 |
| opal.version_compare . . . . .   | 169 |
| opal.vocabularies . . . . .      | 170 |
| opal.vocabulary . . . . .        | 171 |
| opal.workspaces . . . . .        | 172 |
| opal.workspace_restore . . . . . | 172 |
| opal.workspace_rm . . . . .      | 173 |
| opal.workspace_save . . . . .    | 174 |

|              |            |
|--------------|------------|
| <b>Index</b> | <b>175</b> |
|--------------|------------|

---

|                     |   |
|---------------------|---|
| dictionary.annotate | <i>Set variable annotation with a taxonomy term</i> |
|---------------------|---|

---

## 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 | <i>Apply the dictionary to a tibble</i> |
|------------------|---|

---

### 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 | <i>Get DataSHIELD activity</i> |
|------------------|--------------------------------|

---

## 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

|                      |                                       |
|----------------------|---------------------------------------|
| <code>opal</code>    | Opal connection object.               |
| <code>user</code>    | Optional user name.                   |
| <code>profile</code> | Optional profile name.                |
| <code>from</code>    | Optional start date.                  |
| <code>to</code>      | Optional end date.                    |
| <code>df</code>      | 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 | <i>Get a DataSHIELD method</i> |
|--------------------|--------------------------------|

---

## 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 <a href="#">dsadmin.profiles</a> . |

## See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.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 <a href="#">dsadmin.profiles</a> . |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.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 <a href="#">dsadmin.profiles</a> . |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.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 <a href="#">dsadmin.profiles</a> . |

**Value**

TRUE if installed

## See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.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 <a href="#">dsadmin.profiles</a> . |

**Value**

TRUE if installed

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.install_github_package(o, 'ds0mics', 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 <a href="#">dsadmin.profiles</a> . |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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')
# 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 <a href="#">dsadmin.profiles</a> .                                   |

## Value

TRUE if installed

## See Also

Other DataSHIELD functions: `dsadmin.get_methods()`, `dsadmin.get_method()`, `dsadmin.get_options()`, `dsadmin.install_github_package()`, `dsadmin.install_local_package()`, `dsadmin.installed_package()`, `dsadmin.package_descriptions()`, `dsadmin.package_description()`, `dsadmin.publish_package()`, `dsadmin.remove_package()`, `dsadmin.rm_methods()`, `dsadmin.rm_method()`, `dsadmin.rm_options()`, `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.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

|                   |                                      |
|-------------------|--------------------------------------|
| <code>opal</code> | Opal connection object.              |
| <code>all</code>  | 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, 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
- 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**

|                      |  |
|----------------------|--|
| <code>opal</code>    | Opal object or list of opal objects.   |
| <code>pkg</code>     | Package name.  |
| <code>fields</code>  | 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. |
| <code>profile</code> | The DataSHIELD profile name to which operation applies. See also <a href="#">dsadmin.profiles</a> .  |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.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**

|                      |  |
|----------------------|--|
| <code>opal</code>    | Opal object or list of opal objects.   |
| <code>fields</code>  | 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. |
| <code>df</code>      | Return a data.frame (default is TRUE)  |
| <code>profile</code> | The DataSHIELD profile name to which operation applies. See also <a href="#">dsadmin.profiles</a> .  |

**Value**

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

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.package_descriptions(o)  
opal.logout(o)  
  
## End(Not run)
```

---

|              |                                       |
|--------------|---------------------------------------|
| dsadmin.perm | <i>Get the DataSHIELD permissions</i> |
|--------------|---------------------------------------|

---

**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 | <i>Add or update a DataSHIELD permission</i> |
|------------------|--|

---

**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 | <i>Get a DataSHIELD profile</i> |
|-----------------|---------------------------------|

---

### 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\\_add\(\)](#), [dsadmin.profile\\_perm\\_delete\(\)](#), [dsadmin.profile\\_perm\(\)](#), [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 | <i>Get DataSHIELD profiles</i> |
|------------------|--------------------------------|

---

### 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\\_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\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profile\(\)](#)

**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\\_create\(\)](#), [dsadmin.profile\\_delete\(\)](#), [dsadmin.profile\\_enable\(\)](#), [dsadmin.profile\\_exists\(\)](#), [dsadmin.profile\\_init\(\)](#), [dsadmin.profile\\_perm\\_add\(\)](#), [dsadmin.profile\\_perm\\_delete\(\)](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

**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\\_access\(\)](#), [dsadmin.profile\\_delete\(\)](#), [dsadmin.profile\\_enable\(\)](#), [dsadmin.profile\\_exists\(\)](#), [dsadmin.profile\\_init\(\)](#), [dsadmin.profile\\_perm\\_add\(\)](#), [dsadmin.profile\\_perm\\_delete\(\)](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

## 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\\_access\(\)](#), [dsadmin.profile\\_create\(\)](#), [dsadmin.profile\\_enable\(\)](#), [dsadmin.profile\\_exists\(\)](#), [dsadmin.profile\\_init\(\)](#), [dsadmin.profile\\_perm\\_add\(\)](#), [dsadmin.profile\\_perm\\_delete\(\)](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

## 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**

|                      |                        |
|----------------------|------------------------|
| <code>opal</code>    | Opal object.           |
| <code>name</code>    | Name of the profile.   |
| <code>enabled</code> | Default value is TRUE. |

**See Also**

Other DataSHIELD profiles: [dsadmin.profile\\_access\(\)](#), [dsadmin.profile\\_create\(\)](#), [dsadmin.profile\\_delete\(\)](#), [dsadmin.profile\\_exists\(\)](#), [dsadmin.profile\\_init\(\)](#), [dsadmin.profile\\_perm\\_add\(\)](#), [dsadmin.profile\\_perm\\_del](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

**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\\_access\(\)](#), [dsadmin.profile\\_create\(\)](#), [dsadmin.profile\\_delete\(\)](#), [dsadmin.profile\\_enable\(\)](#), [dsadmin.profile\\_init\(\)](#), [dsadmin.profile\\_perm\\_add\(\)](#), [dsadmin.profile\\_perm\\_delete\(\)](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

**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, optionally 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\\_access\(\)](#), [dsadmin.profile\\_create\(\)](#), [dsadmin.profile\\_delete\(\)](#), [dsadmin.profile\\_enable\(\)](#), [dsadmin.profile\\_exists\(\)](#), [dsadmin.profile\\_perm\\_add\(\)](#), [dsadmin.profile\\_perm\\_delete\(\)](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

## 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\\_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\(\)](#), [dsadmin.profile\(\)](#)

## 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

|                         |   |
|-------------------------|---|
| <code>opal</code>       | Opal connection object.   |
| <code>name</code>       | Profile.  |
| <code>subject</code>    | A vector of subject identifiers: user names or group names (depending on the type). |
| <code>type</code>       | The type of subject: user (default) or group.                                       |
| <code>permission</code> | The permission to apply: use.   |

### 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\\_delete\(\)](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

### 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

|                      |   |
|----------------------|---|
| <code>opal</code>    | Opal connection object.   |
| <code>name</code>    | Profile name.   |
| <code>subject</code> | A vector of subject identifiers: user names or group names (depending on the type). |
| <code>type</code>    | The type of subject: user (default) or group.                                       |

## 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\\_add\(\)](#), [dsadmin.profile\\_perm\(\)](#), [dsadmin.profile\\_rparser\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

## 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\\_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\\_perm\(\)](#), [dsadmin.profiles\(\)](#), [dsadmin.profile\(\)](#)

## 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

|                      |   |
|----------------------|---|
| <code>opal</code>    | Opal object or list of opal objects.  |
| <code>pkg</code>     | Package name.   |
| <code>profile</code> | The DataSHIELD profile name to which operation applies. See also <a href="#">dsadmin.profiles</a> . |

## Value

TRUE if successful

## See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.publish_package(o, 'dsBase')
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 <a href="#">dsadmin.profiles</a> . |

### See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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.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 <a href="#">dsadmin.profiles</a> . |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_options\(\)](#), [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_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 <a href="#">dsadmin.profiles</a> . |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [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_methods(o)
opal.logout(o)

## End(Not run)
```

---

|                   |                                   |
|-------------------|-----------------------------------|
| dsadmin.rm_option | <i>Remove a DataSHIELD option</i> |
|-------------------|-----------------------------------|

---

**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 <a href="#">dsadmin.profiles</a> . |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [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 | <i>Remove all DataSHIELD options</i> |
|--------------------|--------------------------------------|

---

## 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 <a href="#">dsadmin.profiles</a> . |

## See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_option\(\)](#), [dsadmin.rm\\_package\\_methods\(\)](#), [dsadmin.set\\_method\(\)](#), [dsadmin.set\\_option\(\)](#), [dsadmin.set\\_package\\_methods\(\)](#), [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 <a href="#">dsadmin.profiles</a> . |

### See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [dsadmin.rm\\_option\(\)](#), [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 <a href="#">dsadmin.profiles</a> . |

**See Also**

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [dsadmin.rm\\_option\(\)](#), [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 <a href="#">dsadmin.profiles</a> . |

### See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [dsadmin.rm\\_option\(\)](#), [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

|                      |   |
|----------------------|---|
| <code>opal</code>    | Opal object or list of opal objects.  |
| <code>pkg</code>     | Package name.   |
| <code>type</code>    | Type of the method: "aggregate" or "assign". Default is NULL (=all type of methods).                |
| <code>profile</code> | The DataSHIELD profile name to which operation applies. See also <a href="#">dsadmin.profiles</a> . |

## Value

TRUE if successfull

## See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [dsadmin.rm\\_option\(\)](#), [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

|                      |   |
|----------------------|---|
| <code>opal</code>    | Opal object or list of opal objects.  |
| <code>pkg</code>     | Package name.   |
| <code>profile</code> | The DataSHIELD profile name to which operation applies. See also <a href="#">dsadmin.profiles</a> . |

### Value

TRUE if successfull

### See Also

Other DataSHIELD functions: [dsadmin.get\\_methods\(\)](#), [dsadmin.get\\_method\(\)](#), [dsadmin.get\\_options\(\)](#), [dsadmin.install\\_github\\_package\(\)](#), [dsadmin.install\\_local\\_package\(\)](#), [dsadmin.install\\_package\(\)](#), [dsadmin.installed\\_package\(\)](#), [dsadmin.package\\_descriptions\(\)](#), [dsadmin.package\\_description\(\)](#), [dsadmin.publish\\_package\(\)](#), [dsadmin.remove\\_package\(\)](#), [dsadmin.rm\\_methods\(\)](#), [dsadmin.rm\\_method\(\)](#), [dsadmin.rm\\_options\(\)](#), [dsadmin.rm\\_option\(\)](#), [dsadmin.rm\\_package\\_methods\(\)](#), [dsadmin.set\\_method\(\)](#), [dsadmin.set\\_option\(\)](#), [dsadmin.set\\_package\\_methods\(\)](#)

### Examples

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

---

|                 |                       |
|-----------------|-----------------------|
| oadmin.activity | <i>Get R activity</i> |
|-----------------|-----------------------|

---

## 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\\_packages\(\)](#), [oadmin.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#), [oadmin.remove\\_package\(\)](#)

**Examples**

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

---

```
oadmin.installed_package
```

*Check package is installed*

---

**Description**

Check package is installed

**Usage**

```
oadmin.installed_package(opal, pkg, 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 <a href="#">opal.profiles</a> . |

**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 <a href="#">opal.profiles</a> . |

**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 <a href="#">opal.profiles</a> . |

## 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\\_packages\(\)](#), [oadmin.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#), [oadmin.remove\\_package\(\)](#)

## Examples

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

## End(Not run)
```

oadmin.install\_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 <a href="#">opal.profiles</a> . |

### See Also

Other administration functions: [oadmin.install\\_bioconductor\\_package\(\)](#), [oadmin.install\\_devtools\(\)](#), [oadmin.install\\_github\\_package\(\)](#), [oadmin.install\\_local\\_package\(\)](#), [oadmin.install\\_package\(\)](#), [oadmin.installed\\_devtools\(\)](#), [oadmin.installed\\_packages\(\)](#), [oadmin.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#), [oadmin.remove\\_package\(\)](#)

### Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
oadmin.install_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\\_packages\(\)](#), [oadmin.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#), [oadmin.remove\\_package\(\)](#)

**Examples**

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

## End(Not run)
```

---

```
oadmin.install_github_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\\_packages\(\)](#), [oadmin.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#), [oadmin.remove\\_package\(\)](#)

## Examples

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

## End(Not run)
```

---

oadmin.install\_local\_package

*Install a package from a local archive file*

---

## Description

Install a package from a package archive file. This will upload the archive file and run its installation in the R server. 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 <a href="#">opal.profiles</a> . |

## 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\\_packages\(\)](#), [oadmin.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#), [oadmin.remove\\_package\(\)](#)

## Examples

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

## End(Not run)
```

---

oadmin.install\_package

*Install 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 <a href="#">opal.profiles</a> . |

## 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\\_packages\(\)](#), [oadmin.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#), [oadmin.remove\\_package\(\)](#)

**Examples**

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

## End(Not run)
```

---

|            |                           |
|------------|---------------------------|
| oadmin.log | <i>Get Opal main logs</i> |
|------------|---------------------------|

---

**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 | <i>Get Opal REST API logs</i> |
|-----------------|-------------------------------|

---

**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 <a href="#">opal.profiles</a> .  |

## 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.installed\\_package\(\)](#), [oadmin.remove\\_package\(\)](#)

## Examples

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

---

|             |   |
|-------------|---|
| oadmin.perm | <i>Get the R permissions (deprecated)</i> |
|-------------|---|

---

**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 | <i>Add or update a R permission (deprecated)</i> |
|-----------------|--|

---

**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 | <i>Delete a R permission (deprecated)</i> |
|--------------------|---|

---

## 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**

|                      |   |
|----------------------|---|
| <code>opal</code>    | Opal object or list of opal objects.  |
| <code>pkg</code>     | Package name.   |
| <code>profile</code> | The R servers profile name to which operation applies. See also <a href="#">opal.profiles</a> . |

**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.installed\\_package\(\)](#), [oadmin.package\\_description\(\)](#)

**Examples**

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

---

`oadmin.r_perm` *Get the R permissions*

---

**Description**

Get the permissions that were applied to the R service.

**Usage**

```
oadmin.r_perm(opal)
```

**Arguments**

|                   |                         |
|-------------------|-------------------------|
| <code>opal</code> | 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 | <i>Add or update a R permission</i> |
|-------------------|-------------------------------------|

---

**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 EPOCH 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**

|                         |   |
|-------------------------|---|
| <code>opal</code>       | Opal connection object.   |
| <code>subject</code>    | A vector of subject identifiers: user names or group names (depending on the type). |
| <code>type</code>       | The type of subject: user (default) or group.                                       |
| <code>permission</code> | The permission to apply: <code>project_add</code> or <code>administrate</code> .    |

**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 | <i>Add a user</i> |
|-----------------|-------------------|

---

**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 | <i>Delete a user</i> |
|--------------------|----------------------|

---

**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 | <i>Enable a user</i> |
|--------------------|----------------------|

---

**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 | <i>Check user exists</i> |
|--------------------|--------------------------|

---

## 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 |
| <i>Reset user password</i> |

---

**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 | <i>Apply the annotations to a Opal table</i> |
|---------------|--|

---

**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 exist 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 or NA the annotation is removed). |

**See Also**

Other datasource functions: [opal.annotations\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

**Examples**

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

## End(Not run)
```

---

|                  |  |
|------------------|--|
| opal.annotations | <i>Get the annotations of a Opal table</i> |
|------------------|--|

---

## Description

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

## Usage

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

## Arguments

|            |                                      |
|------------|--------------------------------------|
| opal       | Opal object.                         |
| datasource | Name of the datasource.              |
| table      | Name of the table in the datasource. |

## Value

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

## See Also

Other datasource functions: [opal.annotate\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

## Examples

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

---

|             |                                      |
|-------------|--------------------------------------|
| opal.assign | <i>Data or expression assignment</i> |
|-------------|--------------------------------------|

---

### 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 refer to a table). See javascript documentation: <a href="http://wiki.obiba.org/display/OPALDOC/Variable+Methods">http://wiki.obiba.org/display/OPALDOC/Variable+Methods</a> |
| missings     | If TRUE, missing values will be pushed from Opal to R, default is FALSE. Ignored if value is an R expression.   |
| identifiers  | Name of the identifiers mapping to use when assigning entities to R (from Opal 2.0).  |
| id.name      | Add a vector with the given name representing the entity identifiers (from Opal 2.6). Default is NULL.  |
| updated.name | Add a vector with the given name representing the creation and last update timestamps (from Opal 2.6). Default is NULL.   |
| async        | R script is executed asynchronously within the session (default is FALSE). If TRUE, the value returned is the ID of the command to look for (from Opal 2.1).  |

### See Also

Other assignment functions: [opal.assign.data\(\)](#), [opal.assign.resource\(\)](#), [opal.assign.script\(\)](#), [opal.assign.table.tibble\(\)](#), [opal.assign.table\(\)](#)

## Examples

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

## End(Not run)
```

---

|                  |                        |
|------------------|------------------------|
| opal.assign.data | <i>Data assignment</i> |
|------------------|------------------------|

---

## Description

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

## Usage

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

## Arguments

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

## See Also

Other assignment functions: [opal.assign.resource\(\)](#), [opal.assign.script\(\)](#), [opal.assign.table.tibble\(\)](#), [opal.assign.table\(\)](#), [opal.assign\(\)](#)

**Examples**

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

## End(Not run)
```

---

opal.assign.resource    *Resource assignment*

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other assignment functions: [opal.assign.data\(\)](#), [opal.assign.script\(\)](#), [opal.assign.table.tibble\(\)](#), [opal.assign.table\(\)](#), [opal.assign\(\)](#)

**Examples**

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

## End(Not run)
```

---

opal.assign.script      *R script assignment*

---

### Description

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

### Usage

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

### Arguments

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

### See Also

Other assignment functions: [opal.assign.data\(\)](#), [opal.assign.resource\(\)](#), [opal.assign.table.tibble\(\)](#), [opal.assign.table\(\)](#), [opal.assign\(\)](#)

### Examples

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

## End(Not run)
```

---

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

---

### Description

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

**Usage**

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

**Arguments**

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

**See Also**

Other assignment functions: [opal.assign.data\(\)](#), [opal.assign.resource\(\)](#), [opal.assign.script\(\)](#), [opal.assign.table.tibble\(\)](#), [opal.assign\(\)](#)

**Examples**

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

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

## End(Not run)
```

---

```
opal.assign.table.tibble
```

*Data assignment to a tibble*

---

## Description

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

## Usage

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

## Arguments

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

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

### See Also

Other assignment functions: [opal.assign.data\(\)](#), [opal.assign.resource\(\)](#), [opal.assign.script\(\)](#), [opal.assign.table\(\)](#), [opal.assign\(\)](#)

### Examples

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

## End(Not run)
```

---

|                  |                          |
|------------------|--------------------------|
| opal.as_md_table | <i>Array to Markdown</i> |
|------------------|--------------------------|

---

### 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.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

## 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 | <i>Get an asynchronous command</i> |
|--------------|------------------------------------|

---

## Description

Get an asynchronous R commands in the remote R session.

## Usage

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

## Arguments

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

## See Also

Other command functions: [opal.command\\_result\(\)](#), [opal.command\\_rm\(\)](#), [opal.commands\\_rm\(\)](#), [opal.commands\(\)](#)

## Examples

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

## End(Not run)
```

---

|               |                                       |
|---------------|---------------------------------------|
| opal.commands | <i>List the asynchronous commands</i> |
|---------------|---------------------------------------|

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other command functions: [opal.command\\_result\(\)](#), [opal.command\\_rm\(\)](#), [opal.commands\\_rm\(\)](#), [opal.command\(\)](#)

**Examples**

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

---

|                  |   |
|------------------|---|
| opal.commands_rm | <i>Remove all asynchronous commands</i> |
|------------------|---|

---

**Description**

Remove all asynchronous R commands in the remote R session.

**Usage**

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

**Arguments**

|      |              |
|------|--------------|
| opal | Opal object. |
|------|--------------|

**See Also**

Other command functions: [opal.command\\_result\(\)](#), [opal.command\\_rm\(\)](#), [opal.commands\(\)](#), [opal.command\(\)](#)

**Examples**

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

## End(Not run)
```

---

`opal.command_result`    *Get result of an asynchronous command*

---

**Description**

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

**Usage**

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

**Arguments**

|                   |                                   |
|-------------------|-----------------------------------|
| <code>opal</code> | Opal object.                      |
| <code>id</code>   | R command ID.                     |
| <code>wait</code> | Wait for the command to complete. |

**See Also**

Other command functions: [opal.command\\_rm\(\)](#), [opal.commands\\_rm\(\)](#), [opal.commands\(\)](#), [opal.command\(\)](#)

**Examples**

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

## End(Not run)
```

---

|                 |                                       |
|-----------------|---------------------------------------|
| opal.command_rm | <i>Remove an asynchronous command</i> |
|-----------------|---------------------------------------|

---

**Description**

Remove an asynchronous R commands in the remote R session.

**Usage**

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

**Arguments**

|      |               |
|------|---------------|
| opal | Opal object.  |
| id   | R command ID. |

**See Also**

Other command functions: [opal.command\\_result\(\)](#), [opal.commands\\_rm\(\)](#), [opal.commands\(\)](#), [opal.command\(\)](#)

**Examples**

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

---

|                 |                         |
|-----------------|-------------------------|
| opal.datasource | <i>Get a datasource</i> |
|-----------------|-------------------------|

---

**Description**

Get a datasource

**Usage**

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

**Arguments**

|            |                         |
|------------|-------------------------|
| opal       | Opal object.            |
| datasource | Name of the datasource. |

### See Also

Other datasource functions: [opal.annotate\(\)](#), [opal.annotations\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasources\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

### 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 | <i>Get datasources</i> |
|------------------|------------------------|

---

### Description

Get datasources

### Usage

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

### Arguments

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

### See Also

Other datasource functions: [opal.annotate\(\)](#), [opal.annotations\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

### Examples

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

## End(Not run)
```

---

|             |  |
|-------------|--|
| opal.delete | <i>Generic REST resource deletion.</i> |
|-------------|--|

---

**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 | <i>Execute a R script</i> |
|--------------|---------------------------|

---

**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 | <i>Get file content</i> |
|-----------|-------------------------|

---

## Description

Get file content from the Opal file system.

## Usage

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

## Arguments

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

## See Also

Other file functions: [opal.file\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_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\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

### Examples

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

## End(Not run)
```

---

opal.file\_download    *Download a file*

---

### Description

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

### Usage

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

**Arguments**

|             |   |
|-------------|---|
| opal        | Opal object.  |
| source      | Path to the file in the Opal file system.   |
| destination | Path to the file to be written. If 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\\_cp\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

**Examples**

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

## End(Not run)
```

---

|              |                                 |
|--------------|---------------------------------|
| opal.file_ls | <i>List content of a folder</i> |
|--------------|---------------------------------|

---

**Description**

List content of a folder in the Opal file system.

**Usage**

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

**Arguments**

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

### See Also

Other file functions: [opal.file\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

### Examples

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

## End(Not run)
```

---

|                 |                      |
|-----------------|----------------------|
| opal.file_mkdir | <i>Make a folder</i> |
|-----------------|----------------------|

---

### Description

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

### Usage

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

### Arguments

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

### See Also

Other file functions: [opal.file\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

### Examples

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

## End(Not run)
```

---

opal.file\_mkdir\_tmp     *Make a temporary folder*

---

**Description**

Make a 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\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

**Examples**

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

## End(Not run)
```

---

opal.file\_mv             *Move and/or rename a file*

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other file functions: [opal.file\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

**Examples**

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

## End(Not run)
```

---

|                |                    |
|----------------|--------------------|
| opal.file_read | <i>Read a file</i> |
|----------------|--------------------|

---

**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 exist).                         |
| destination | Path to the destination file or folder. Any required sub-folders will be created. |

**See Also**

Other file functions: [opal.file\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

**Examples**

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

## End(Not run)
```

---

|              |                      |
|--------------|----------------------|
| opal.file_rm | <i>Remove a file</i> |
|--------------|----------------------|

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other file functions: [opal.file\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

**Examples**

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

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

## End(Not run)
```

---

opal.file\_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\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

### 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 | <i>Upload a file or a folder</i> |
|------------------|----------------------------------|

---

### 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\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_write\(\)](#), [opal.file\(\)](#)

### 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 | <i>Write a file</i> |
|-----------------|---------------------|

---

## 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 exist 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 omitted, file with same name will be written. |

## See Also

Other file functions: [opal.file\\_cp\(\)](#), [opal.file\\_download\(\)](#), [opal.file\\_ls\(\)](#), [opal.file\\_mkdir\\_tmp\(\)](#), [opal.file\\_mkdir\(\)](#), [opal.file\\_mv\(\)](#), [opal.file\\_read\(\)](#), [opal.file\\_rm\(\)](#), [opal.file\\_unzip\(\)](#), [opal.file\\_upload\(\)](#), [opal.file\(\)](#)

## Examples

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

## End(Not run)
```

---

opal.get                      *Generic REST resource getter.*

---

### Description

Generic REST resource getter.

### Usage

```
opal.get(  
  opal,  
  ...,  
  query = list(),  
  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.source\(\)](#), [opal.execute\(\)](#), [opal.unload\\_package\(\)](#)

### Examples

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

---

opal.login      *Opal login*

---

### Description

Log in Opal(s). 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
)
```

**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 htrr (call htrr::htrr_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 <a href="#">opal.profiles</a> . |
| restore  | Workspace ID to be restored (see also opal.logout)   |

**Value**

A opal object or a list of opal objects.

**See Also**

Other connection functions: [opal.logout\(\)](#), [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 = 'HYG16L00VaX400UardNbiqmr2ByBpRke',
               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 = 'HYG16L00VaX400UardNbiqmr2ByBpRke',
        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 | <i>Logout from Opal(s)</i> |
|-------------|----------------------------|

---

### 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 | <i>Get the permissions of a subject</i> |
|------------|---|

---

**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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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 | <i>List R profiles</i> |
|---------------|------------------------|

---

**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 | <i>Get a project</i> |
|--------------|----------------------|

---

**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\\_databases\(\)](#), [opal.projects\(\)](#)

**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 | <i>Get projects</i> |
|---------------|---------------------|

---

**Description**

Get projects

**Usage**

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

**Arguments**

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

**See Also**

Other project functions: [opal.project\\_create\(\)](#), [opal.project\\_delete\(\)](#), [opal.project\\_exists\(\)](#), [opal.projects\\_databases\(\)](#), [opal.project\(\)](#)

**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\\_create\(\)](#), [opal.project\\_delete\(\)](#), [opal.project\\_exists\(\)](#), [opal.projects\(\)](#), [opal.project\(\)](#)

### 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
)
```

**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, 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). |

**See Also**

Other project functions: [opal.project\\_delete\(\)](#), [opal.project\\_exists\(\)](#), [opal.projects\\_databases\(\)](#), [opal.projects\(\)](#), [opal.project\(\)](#)

**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 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\\_create\(\)](#), [opal.project\\_exists\(\)](#), [opal.projects\\_databases\(\)](#), [opal.projects\(\)](#), [opal.project\(\)](#)

### Examples

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

## End(Not run)
```

---

opal.project\_exists    *Check a project exists*

---

### Description

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

### Usage

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

### Arguments

|         |                     |
|---------|---------------------|
| opal    | Opal object.        |
| project | Name of the project |

### Value

A logical

### See Also

Other project functions: [opal.project\\_create\(\)](#), [opal.project\\_delete\(\)](#), [opal.projects\\_databases\(\)](#), [opal.projects\(\)](#), [opal.project\(\)](#)

### 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-rsript",
  callback = NULL
)
```

## Arguments

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

## See Also

Other REST functions: [opal.delete\(\)](#), [opal.get\(\)](#), [opal.post\(\)](#)

## Examples

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

## End(Not run)
```

---

opal.report

*Opal report*

---

## Description

Helper function for generating reports.

## Usage

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

## Arguments

|            |  |
|------------|--|
| input      | Path to the R markdown report file   |
| output     | Directory path where to output 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 | <i>Get a resource reference of a project</i> |
|---------------|--|

---

**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\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#)

**Examples**

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

---

|                |   |
|----------------|---|
| opal.resources | <i>Get the resource references of a project</i> |
|----------------|---|

---

**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\\_create\(\)](#), [opal.resource\\_delete\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\\_remove\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\\_remove\(\)](#), [opal.resource\(\)](#)

**Examples**

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

## End(Not run)
```

---

`opal.resources_perm` *Get the permissions on any resource*

---

**Description**

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

**Usage**

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

**Arguments**

|         |                         |
|---------|-------------------------|
| opal    | Opal connection object. |
| project | The project name.       |

**Value**

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

**See Also**

Other resource functions: [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.resource\\_perm\\_remove\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\\_remove\(\)](#), [opal.resource\(\)](#)

**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\\_create\(\)](#), [opal.resource\\_delete\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

**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

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

### See Also

Other resource functions: [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.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

### 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\\_delete\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

## Examples

```
## Not run:
o <- opal.login('administrator','password', url='https://opal-demo.obiba.org')
opal.resource_create(o, 'RSRC', 'CNSIM4',
  url = 'opal+https://opal-demo.obiba.org/ws/files/projects/RSRC/CNSIM3.zip',
  format = 'csv', secret = 'EeTtQGIob6hai05bx6FUfVvIGkeZJfGq')
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\\_create\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

## 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\\_create\(\)](#), [opal.resource\\_delete\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

## Examples

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

## End(Not run)
```

---

`opal.resource_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**

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

**See Also**

Other resource functions: [opal.resource\\_create\(\)](#), [opal.resource\\_delete\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

## 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\\_create\(\)](#), [opal.resource\\_delete\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

## Examples

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

## End(Not run)
```

---

opal.resource\_perm      *Get the permissions on a resource*

---

## Description

Get the permissions that were applied on a resource.

## Usage

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

## Arguments

|          |                         |
|----------|-------------------------|
| opal     | Opal connection object. |
| project  | The project name.       |
| resource | The resource name.      |

## Value

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

## See Also

Other resource functions: [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\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

## 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\\_create\(\)](#), [opal.resource\\_delete\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_delete\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

**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\\_create\(\)](#), [opal.resource\\_delete\(\)](#), [opal.resource\\_exists\(\)](#), [opal.resource\\_extension\\_create\(\)](#), [opal.resource\\_get\(\)](#), [opal.resource\\_perm\\_add\(\)](#), [opal.resource\\_perm\(\)](#), [opal.resources\\_perm\\_add\(\)](#), [opal.resources\\_perm\\_delete\(\)](#), [opal.resources\\_perm\(\)](#), [opal.resources\(\)](#), [opal.resource\(\)](#)

**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

|                       |  |
|-----------------------|--|
| <code>opal</code>     | Opal connection object.  |
| <code>project</code>  | Project name where the view will be located.   |
| <code>table</code>    | View name to be created.   |
| <code>resource</code> | Fully qualified resource name.   |
| <code>type</code>     | Entity type, default is "Participant".   |
| <code>idColumn</code> | Name of the column which contains the entity identifiers. If not specified, the first column will be used.   |
| <code>profile</code>  | R server profile to use for establishing the connection with the resource. If not specified, the profile will be 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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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 | <i>Remove a R symbol (deprecated)</i> |
|---------|---------------------------------------|

---

**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.sql | <i>Execute a SQL query on tables</i> |
|----------|--------------------------------------|

---

**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 | <i>Remove a R symbol</i> |
|----------------|--------------------------|

---

### Description

Remove a symbol from the remote R session.

### Usage

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

### Arguments

|        |                       |
|--------|-----------------------|
| opal   | Opal object.          |
| symbol | Name of the R symbol. |

### See Also

Other symbol functions: [opal.rm\(\)](#), [opal.symbol\\_import\(\)](#), [opal.symbol\\_save\(\)](#), [opal.symbols\(\)](#)

### Examples

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

## End(Not run)
```

---

opal.symbol\_save      *Save a tibble*

---

### Description

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

### Usage

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

### Arguments

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

### See Also

Other symbol functions: [opal.rm\(\)](#), [opal.symbol\\_import\(\)](#), [opal.symbol\\_rm\(\)](#), [opal.symbols\(\)](#)

### Examples

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

## End(Not run)
```

---

opal.table      *Get a table of a datasource*

---

### Description

Get a table of a datasource

### Usage

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

**Arguments**

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

**See Also**

Other datasource functions: [opal.annotate\(\)](#), [opal.annotations\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

**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 | <i>Get tables of a datasource</i> |
|-------------|-----------------------------------|

---

**Description**

Get tables of a datasource

**Usage**

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

**Arguments**

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

**See Also**

Other datasource functions: [opal.annotate\(\)](#), [opal.annotations\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasources\(\)](#), [opal.datasource\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

## 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 | <i>Get the permissions on any table of a project</i> |
|------------------|--|

---

## 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 | <i>Create an Opal table or view</i> |
|-------------------|-------------------------------------|

---

**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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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 | <i>Delete a Opal table</i> |
|-------------------|----------------------------|

---

**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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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
)
```

**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. |

**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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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 <code>base::readRDS()</code> , while other formats are supported by the <code>haven</code> 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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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: <a href="http://wiki.obiba.org/display/OPALDOC/Variable+Methods">http://wiki.obiba.org/display/OPALDOC/Variable+Methods</a> |
| 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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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_add()`, `opal.table_perm_delete()`, `opal.table_perm()`, `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)
```

---

|                              |                                       |
|------------------------------|---------------------------------------|
| <code>opal.table_perm</code> | <i>Get the permissions on a table</i> |
|------------------------------|---------------------------------------|

---

**Description**

Get the permissions that were applied on a table.

**Usage**

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

**Arguments**

|                      |   |
|----------------------|---|
| <code>opal</code>    | Opal connection object.                       |
| <code>project</code> | Project name where the table will be located. |
| <code>table</code>   | 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\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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\\_add\(\)](#), [opal.table\\_perm\(\)](#), [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 | <i>Save a local tibble as a Opal table</i> |
|-----------------|--|

---

### 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 <a href="#">opal.table_truncate</a> 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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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\\_add\(\)](#), [opal.table\\_perm\\_delete\(\)](#), [opal.table\\_perm\(\)](#), [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 | <i>Get a task</i> |
|-----------|-------------------|

---

## Description

Get the details of a specific task.

## Usage

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

## Arguments

|      |                  |
|------|------------------|
| opal | Opal object.     |
| id   | Task identifier. |

## See Also

Other task functions: [opal.task\\_cancel\(\)](#), [opal.task\\_wait\(\)](#), [opal.tasks\(\)](#)

## Examples

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

## End(Not run)
```

---

|            |                      |
|------------|----------------------|
| opal.tasks | <i>Get the tasks</i> |
|------------|----------------------|

---

### Description

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

### Usage

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

### Arguments

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

### See Also

Other task functions: [opal.task\\_cancel\(\)](#), [opal.task\\_wait\(\)](#), [opal.task\(\)](#)

### Examples

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

---

|                  |                      |
|------------------|----------------------|
| opal.task_cancel | <i>Cancel a task</i> |
|------------------|----------------------|

---

### Description

Tries to cancel a task.

### Usage

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

### Arguments

|      |                  |
|------|------------------|
| opal | Opal object.     |
| id   | Task identifier. |

**See Also**

Other task functions: [opal.task\\_wait\(\)](#), [opal.tasks\(\)](#), [opal.task\(\)](#)

**Examples**

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

## End(Not run)
```

---

|                |                                     |
|----------------|-------------------------------------|
| opal.task_wait | <i>Wait for a task to complete.</i> |
|----------------|-------------------------------------|

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other task functions: [opal.task\\_cancel\(\)](#), [opal.tasks\(\)](#), [opal.task\(\)](#)

**Examples**

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

## End(Not run)
```

---

|                 |                       |
|-----------------|-----------------------|
| opal.taxonomies | <i>Get taxonomies</i> |
|-----------------|-----------------------|

---

### Description

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

### Usage

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

### Arguments

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

### See Also

Other taxonomy functions: [opal.taxonomy\\_delete\(\)](#), [opal.taxonomy\\_download\(\)](#), [opal.taxonomy\\_upload\(\)](#), [opal.taxonomy\(\)](#), [opal.terms\(\)](#), [opal.vocabularies\(\)](#), [opal.vocabulary\(\)](#)

### Examples

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

---

|               |                       |
|---------------|-----------------------|
| opal.taxonomy | <i>Get a taxonomy</i> |
|---------------|-----------------------|

---

### Description

Get a specific taxonomy details.

### Usage

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

### Arguments

|          |                       |
|----------|-----------------------|
| opal     | Opal object.          |
| taxonomy | Name of the taxonomy. |

**See Also**

Other taxonomy functions: [opal.taxonomies\(\)](#), [opal.taxonomy\\_delete\(\)](#), [opal.taxonomy\\_download\(\)](#), [opal.taxonomy\\_upload\(\)](#), [opal.terms\(\)](#), [opal.vocabularies\(\)](#), [opal.vocabulary\(\)](#)

**Examples**

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

## End(Not run)
```

---

opal.taxonomy\_delete    *Delete a taxonomy*

---

**Description**

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

**Usage**

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

**Arguments**

|          |                       |
|----------|-----------------------|
| opal     | Opal object.          |
| taxonomy | Name of the taxonomy. |

**See Also**

Other taxonomy functions: [opal.taxonomies\(\)](#), [opal.taxonomy\\_download\(\)](#), [opal.taxonomy\\_upload\(\)](#), [opal.taxonomy\(\)](#), [opal.terms\(\)](#), [opal.vocabularies\(\)](#), [opal.vocabulary\(\)](#)

**Examples**

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

## End(Not run)
```

opal.taxonomy\_download

*Download a taxonomy file*

---

### Description

Download a taxonomy stored in a file in YAML format.

### Usage

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

### Arguments

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

### See Also

Other taxonomy functions: [opal.taxonomies\(\)](#), [opal.taxonomy\\_delete\(\)](#), [opal.taxonomy\\_upload\(\)](#), [opal.taxonomy\(\)](#), [opal.terms\(\)](#), [opal.vocabularies\(\)](#), [opal.vocabulary\(\)](#)

### Examples

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

---

opal.taxonomy\_upload *Upload a taxonomy file*

---

### Description

Upload a taxonomy stored in a local file in YAML format. This operation will fail if the taxonomy already exists, see [opal.taxonomy\\_delete](#).

### Usage

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

**Arguments**

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

**See Also**

Other taxonomy functions: [opal.taxonomies\(\)](#), [opal.taxonomy\\_delete\(\)](#), [opal.taxonomy\\_download\(\)](#), [opal.taxonomy\\_upload\(\)](#), [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 | <i>Get the terms of a vocabulary</i> |
|------------|--------------------------------------|

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other taxonomy functions: [opal.taxonomies\(\)](#), [opal.taxonomy\\_delete\(\)](#), [opal.taxonomy\\_download\(\)](#), [opal.taxonomy\\_upload\(\)](#), [opal.taxonomy\(\)](#), [opal.vocabularies\(\)](#), [opal.vocabulary\(\)](#)

## Examples

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

## End(Not run)
```

---

|            |                                    |
|------------|------------------------------------|
| opal.token | <i>Get a personal access token</i> |
|------------|------------------------------------|

---

## 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 | <i>Get the personal access tokens</i> |
|-------------|---------------------------------------|

---

### 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\\_datashield\\_create\(\)](#), [opal.token\\_delete\(\)](#), [opal.token\\_r\\_create\(\)](#), [opal.token\\_renew\(\)](#), [opal.token\\_sql\\_create\(\)](#), [opal.token\(\)](#)

### 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 | <i>Create a personal access token for Datashield usage</i> |
|------------------------------|--|

---

### 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\\_delete\(\)](#), [opal.token\\_r\\_create\(\)](#), [opal.token\\_renew\(\)](#), [opal.token\\_sql\\_create\(\)](#), [opal.tokens\(\)](#), [opal.token\(\)](#)

**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\\_datashield\\_create\(\)](#), [opal.token\\_r\\_create\(\)](#), [opal.token\\_renew\(\)](#), [opal.token\\_sql\\_create\(\)](#), [opal.tokens\(\)](#), [opal.token\(\)](#)

## 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 | <i>Renew an inactive personal access token</i> |
|------------------|--|

---

## 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\\_datashield\\_create\(\)](#), [opal.token\\_delete\(\)](#), [opal.token\\_r\\_create\(\)](#), [opal.token\\_sql\\_create\(\)](#), [opal.tokens\(\)](#), [opal.token\(\)](#)

## 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 be 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\\_datashield\\_create\(\)](#), [opal.token\\_delete\(\)](#), [opal.token\\_renew\(\)](#), [opal.token\\_sql\\_create\(\)](#), [opal.tokens\(\)](#), [opal.token\(\)](#)

### 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\\_datashield\\_create\(\)](#), [opal.token\\_delete\(\)](#), [opal.token\\_r\\_create\(\)](#), [opal.token\\_renew\(\)](#), [opal.tokens\(\)](#), [opal.token\(\)](#)

## 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.source\(\)](#), [opal.execute\(\)](#), [opal.load\\_package\(\)](#)

**Examples**

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

---

opal.valueset     *Get the values of an entity*

---

**Description**

Get the values of an entity in a table.

**Usage**

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

**Arguments**

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

**See Also**

Other datasource functions: [opal.annotate\(\)](#), [opal.annotations\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

**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 | <i>Get a variable of a table</i> |
|---------------|----------------------------------|

---

**Description**

Get a variable of a table

**Usage**

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

**Arguments**

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

**See Also**

Other datasource functions: [opal.annotate\(\)](#), [opal.annotations\(\)](#), [opal.attribute\\_values\(\)](#), [opal.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.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.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variable\\_summary\(\)](#), [opal.variable\(\)](#)

### 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.datasources\(\)](#), [opal.datasource\(\)](#), [opal.tables\(\)](#), [opal.table\(\)](#), [opal.valueset\(\)](#), [opal.variables\(\)](#), [opal.variable\(\)](#)

**Examples**

```
## Not run:
o <- opal.login('administrator', 'password', url='https://opal-demo.obiba.org')
opal.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 | <i>Get the vocabularies of a taxonomy</i> |
|-------------------|---|

---

**Description**

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

**Usage**

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

**Arguments**

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

**See Also**

Other taxonomy functions: [opal.taxonomies\(\)](#), [opal.taxonomy\\_delete\(\)](#), [opal.taxonomy\\_download\(\)](#), [opal.taxonomy\\_upload\(\)](#), [opal.taxonomy\(\)](#), [opal.terms\(\)](#), [opal.vocabulary\(\)](#)

## Examples

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

## End(Not run)
```

---

|                 |                                  |
|-----------------|----------------------------------|
| opal.vocabulary | <i>Get a taxonomy vocabulary</i> |
|-----------------|----------------------------------|

---

## Description

Get a specific vocabulary details.

## Usage

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

## Arguments

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

## See Also

Other taxonomy functions: [opal.taxonomies\(\)](#), [opal.taxonomy\\_delete\(\)](#), [opal.taxonomy\\_download\(\)](#), [opal.taxonomy\\_upload\(\)](#), [opal.taxonomy\(\)](#), [opal.terms\(\)](#), [opal.vocabularies\(\)](#)

## Examples

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

## End(Not run)
```

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

---

### Description

Get the R workspaces from a opal.

### Usage

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

### Arguments

opal                  Opal object.

### See Also

Other workspace functions: [opal.workspace\\_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 | <i>Remove a R workspace from a opal.</i> |
|-------------------|--|

---

**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 workspace. 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`, 36
- `dsadmin.rm_method`, 36
- `dsadmin.rm_methods`, 37
- `dsadmin.rm_option`, 38
- `dsadmin.rm_options`, 39
- `dsadmin.rm_package_methods`, 40
- `dsadmin.set_method`, 41
- `dsadmin.set_option`, 42
- `dsadmin.set_package_methods`, 43
- `dsadmin.unpublish_package`, 44

## \* 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

## \* REST functions

- `opal.delete`, 85
- `opal.get`, 97
- `opal.post`, 102
- `opal.put`, 113

## \* administration functions

- `oadmin.install_bioconductor_package`, 49
- `oadmin.install_cran_package`, 50
- `oadmin.install_devtools`, 50
- `oadmin.install_github_package`, 51
- `oadmin.install_local_package`, 52
- `oadmin.install_package`, 53
- `oadmin.installed_devtools`, 47
- `oadmin.installed_package`, 47
- `oadmin.installed_packages`, 48
- `oadmin.package_description`, 56
- `oadmin.remove_package`, 59

## \* assignment functions

- `opal.assign`, 72
- `opal.assign.data`, 73
- `opal.assign.resource`, 74
- `opal.assign.script`, 75
- `opal.assign.table`, 75
- `opal.assign.table.tibble`, 77

## \* command functions

- `opal.command`, 80
- `opal.command_result`, 82
- `opal.command_rm`, 83
- `opal.commands`, 81
- `opal.commands_rm`, 81

## \* connection functions

- `opal.login`, 98
- `opal.logout`, 100
- `opal.profiles`, 103

## \* datasource functions

- `opal.annotate`, 70
- `opal.annotations`, 71
- `opal.attribute_values`, 79
- `opal.datasource`, 83
- `opal.datasources`, 84
- `opal.table`, 134
- `opal.tables`, 135
- `opal.valueset`, 166

- opal.variable, 167
- opal.variable\_summary, 168
- opal.variables, 168
- \* **execution functions**
  - opal.execute, 85
  - opal.execute.source, 86
  - opal.load\_package, 98
  - opal.unload\_package, 166
- \* **file functions**
  - opal.file, 87
  - opal.file\_cp, 88
  - opal.file\_download, 88
  - opal.file\_ls, 89
  - opal.file\_mkdir, 90
  - opal.file\_mkdir\_tmp, 91
  - opal.file\_mv, 91
  - opal.file\_read, 92
  - opal.file\_rm, 93
  - opal.file\_unzip, 94
  - opal.file\_upload, 95
  - opal.file\_write, 96
- \* **project functions**
  - opal.project, 103
  - opal.project\_create, 107
  - opal.project\_delete, 108
  - opal.project\_exists, 109
  - opal.projects, 104
  - opal.projects\_databases, 105
- \* **resource functions**
  - opal.resource, 115
  - opal.resource\_create, 119
  - opal.resource\_delete, 120
  - opal.resource\_exists, 121
  - opal.resource\_extension\_create, 122
  - opal.resource\_get, 123
  - opal.resource\_perm, 124
  - opal.resource\_perm\_add, 125
  - opal.resource\_perm\_delete, 126
  - opal.resources, 115
  - opal.resources\_perm, 116
  - opal.resources\_perm\_add, 117
  - opal.resources\_perm\_delete, 118
- \* **symbol functions**
  - opal.rm, 129
  - opal.symbol\_import, 132
  - opal.symbol\_rm, 133
  - opal.symbol\_save, 134
  - opal.symbols, 131
- \* **table functions**
  - opal.perms, 101
  - opal.resource\_view\_create, 127
  - opal.resource\_view\_reconnect, 128
  - opal.table\_create, 138
  - opal.table\_delete, 139
  - opal.table\_dictionary\_get, 140
  - opal.table\_dictionary\_update, 140
  - opal.table\_exists, 142
  - opal.table\_export, 143
  - opal.table\_get, 144
  - opal.table\_import, 145
  - opal.table\_perm, 146
  - opal.table\_perm\_add, 147
  - opal.table\_perm\_delete, 148
  - opal.table\_save, 149
  - opal.table\_truncate, 150
  - opal.table\_view\_create, 151
  - opal.table\_view\_update, 152
- \* **task functions**
  - opal.task, 153
  - opal.task\_cancel, 154
  - opal.task\_wait, 155
  - opal.tasks, 154
- \* **taxonomy functions**
  - opal.taxonomies, 156
  - opal.taxonomy, 156
  - opal.taxonomy\_delete, 157
  - opal.taxonomy\_download, 158
  - opal.taxonomy\_upload, 158
  - opal.terms, 159
  - opal.vocabularies, 170
  - opal.vocabulary, 171
- \* **token functions**
  - opal.token, 160
  - opal.token\_datashield\_create, 161
  - opal.token\_delete, 162
  - opal.token\_r\_create, 164
  - opal.token\_renew, 163
  - opal.token\_sql\_create, 165
  - opal.tokens, 161
- \* **user functions**
  - oadmin.user\_add, 65
  - oadmin.user\_delete, 65
  - oadmin.user\_enable, 66
  - oadmin.user\_exists, 67
  - oadmin.user\_profile\_delete, 68

- oadmin.user\_profiles, 68
- oadmin.user\_reset\_password, 69
- oadmin.users, 64
- \* **workspace functions**
  - opal.workspace\_restore, 172
  - opal.workspace\_rm, 173
  - opal.workspace\_save, 174
  - opal.workspaces, 172
  
- 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, 14–17, 19, 21, 22, 35–44
- dsadmin.get\_methods, 13, 14, 15–17, 19, 21, 22, 35–44
- dsadmin.get\_options, 13, 14, 14, 16, 17, 19, 21, 22, 35–44
- dsadmin.install\_github\_package, 13–16, 16, 17, 19, 21, 22, 35–44
- dsadmin.install\_local\_package, 13–17, 17, 19, 21, 22, 35–44
- dsadmin.install\_package, 13–17, 18, 21, 22, 35–44
- dsadmin.installed\_package, 13–15, 15, 17, 19, 21, 22, 35–44
- dsadmin.log, 19
- dsadmin.package\_description, 13–17, 19, 21, 22, 35–44
- dsadmin.package\_descriptions, 13–17, 19, 21, 22, 35–44
- dsadmin.perm, 23
- dsadmin.perm\_add, 23
- dsadmin.perm\_delete, 24
- dsadmin.profile, 25, 26–34
- dsadmin.profile\_access, 25, 26, 26, 27–34
- dsadmin.profile\_create, 25–27, 27, 28–34
- dsadmin.profile\_delete, 25–27, 28, 29–34
- dsadmin.profile\_enable, 25–28, 29, 30–34
- dsadmin.profile\_exists, 25–29, 29, 30–34
- dsadmin.profile\_init, 25–30, 30, 31–34
- dsadmin.profile\_perm, 25–30, 31, 32–34
- dsadmin.profile\_perm\_add, 25–31, 32, 33, 34
- dsadmin.profile\_perm\_delete, 25–32, 33, 34
- dsadmin.profile\_rparser, 25–33, 34
- dsadmin.profiles, 13–18, 21, 22, 25, 25, 27–44
- dsadmin.publish\_package, 13–17, 19, 21, 22, 30, 35, 36–44
- dsadmin.remove\_package, 13–17, 19, 21, 22, 35, 36, 37–44
- dsadmin.rm\_method, 13–17, 19, 21, 22, 35, 36, 36, 38–44
- dsadmin.rm\_methods, 13–17, 19, 21, 22, 35–37, 37, 38–44
- dsadmin.rm\_option, 13–17, 19, 21, 22, 35–38, 38, 39–44
- dsadmin.rm\_options, 13–17, 19, 21, 22, 35–38, 39, 40–44
- dsadmin.rm\_package\_methods, 13–17, 19, 21, 22, 35–39, 40, 41–44
- dsadmin.set\_method, 13–17, 19, 21, 22, 35–40, 41, 42–44
- dsadmin.set\_option, 13–17, 19, 21, 22, 30, 35–41, 42, 43, 44
- dsadmin.set\_package\_methods, 13–17, 19, 21, 22, 30, 35–42, 43, 44
- dsadmin.unpublish\_package, 13–17, 19, 21, 22, 35–43, 44
  
- oadmin.activity, 45
- oadmin.activity\_summary, 46
- oadmin.install\_bioconductor\_package, 47–49, 49, 50–53, 56, 59
- oadmin.install\_cran\_package, 47–49, 50, 51–53, 56, 59
- oadmin.install\_devtools, 47–50, 50, 52, 53, 56, 59
- oadmin.install\_github\_package, 47–51, 51, 52, 53, 56, 59
- oadmin.install\_local\_package, 47–52, 52, 53, 56, 59
- oadmin.install\_package, 47–52, 53, 56, 59
- oadmin.installed\_devtools, 47, 48–53, 56, 59
- oadmin.installed\_package, 47, 47, 49–53, 56, 59
- oadmin.installed\_packages, 47, 48, 48, 49–53, 56, 59
- oadmin.log, 54
- oadmin.log\_rest, 54

- oadmin.log\_sql, [55](#)
- oadmin.package\_description, [47–53](#), [56](#), [59](#)
- oadmin.perm, [57](#)
- oadmin.perm\_add, [57](#)
- oadmin.perm\_delete, [58](#)
- oadmin.r\_perm, [57](#), [59](#)
- oadmin.r\_perm\_add, [57](#), [60](#)
- oadmin.r\_perm\_delete, [58](#), [61](#)
- oadmin.remove\_package, [47–53](#), [56](#), [59](#)
- oadmin.system\_metrics, [61](#)
- oadmin.system\_perm, [62](#)
- oadmin.system\_perm\_add, [63](#)
- oadmin.system\_perm\_delete, [63](#)
- oadmin.user\_add, [64](#), [65](#), [66–69](#)
- oadmin.user\_delete, [64](#), [65](#), [65](#), [66–69](#)
- oadmin.user\_enable, [64–66](#), [66](#), [67–69](#)
- oadmin.user\_exists, [64–66](#), [67](#), [68](#), [69](#)
- oadmin.user\_profile\_delete, [64–68](#), [68](#), [69](#)
- oadmin.user\_profiles, [64–67](#), [68](#), [69](#)
- oadmin.user\_reset\_password, [64–69](#), [69](#)
- oadmin.users, [64](#), [65–69](#)
- opal.annotate, [70](#), [71](#), [79](#), [84](#), [135](#), [167–169](#)
- opal.annotations, [70](#), [71](#), [79](#), [84](#), [135](#), [167–169](#)
- opal.as\_md\_table, [78](#)
- opal.assign, [72](#), [73–76](#), [78](#)
- opal.assign.data, [72](#), [73](#), [74–76](#), [78](#)
- opal.assign.resource, [72](#), [73](#), [74](#), [75](#), [76](#), [78](#)
- opal.assign.script, [72–74](#), [75](#), [76](#), [78](#)
- opal.assign.table, [72–75](#), [75](#), [78](#)
- opal.assign.table.tibble, [72–76](#), [77](#)
- opal.attribute\_values, [70](#), [71](#), [79](#), [84](#), [135](#), [167–169](#)
- opal.command, [80](#), [81–83](#)
- opal.command\_result, [80–82](#), [82](#), [83](#)
- opal.command\_rm, [80–82](#), [83](#)
- opal.commands, [80](#), [81](#), [82](#), [83](#)
- opal.commands\_rm, [80](#), [81](#), [81](#), [82](#), [83](#)
- opal.datasource, [70](#), [71](#), [79](#), [83](#), [84](#), [135](#), [167–169](#)
- opal.datasources, [70](#), [71](#), [79](#), [84](#), [84](#), [135](#), [167–169](#)
- opal.delete, [85](#), [97](#), [102](#), [113](#)
- opal.execute, [85](#), [86](#), [98](#), [166](#)
- opal.execute.source, [86](#), [86](#), [98](#), [166](#)
- opal.file, [87](#), [88–96](#)
- opal.file\_cp, [87](#), [88](#), [89–96](#)
- opal.file\_download, [87](#), [88](#), [88](#), [90–96](#), [143](#)
- opal.file\_ls, [87–89](#), [89](#), [90–96](#)
- opal.file\_mkdir, [87–90](#), [90](#), [91–96](#)
- opal.file\_mkdir\_tmp, [87–90](#), [91](#), [92–96](#)
- opal.file\_mv, [87–91](#), [91](#), [93–96](#)
- opal.file\_read, [87–92](#), [92](#), [93–96](#)
- opal.file\_rm, [87–93](#), [93](#), [94–96](#)
- opal.file\_unzip, [87–93](#), [94](#), [95](#), [96](#)
- opal.file\_upload, [87–94](#), [95](#), [96](#)
- opal.file\_write, [87–95](#), [96](#)
- opal.get, [85](#), [97](#), [102](#), [113](#)
- opal.load\_package, [86](#), [98](#), [166](#)
- opal.login, [98](#), [100](#), [103](#)
- opal.logout, [99](#), [100](#), [103](#)
- opal.perms, [101](#), [127](#), [128](#), [138–144](#), [146–148](#), [150–152](#)
- opal.post, [85](#), [97](#), [102](#), [113](#)
- opal.profiles, [47–53](#), [56](#), [59](#), [99](#), [100](#), [103](#)
- opal.project, [103](#), [104](#), [105](#), [108](#), [109](#)
- opal.project\_backup, [105](#)
- opal.project\_command, [106](#), [106](#), [112](#)
- opal.project\_create, [104](#), [105](#), [107](#), [108](#), [109](#)
- opal.project\_delete, [104](#), [105](#), [108](#), [108](#), [109](#)
- opal.project\_exists, [104](#), [105](#), [108](#), [109](#)
- opal.project\_perm, [110](#)
- opal.project\_perm\_add, [110](#)
- opal.project\_perm\_delete, [111](#)
- opal.project\_restore, [112](#)
- opal.projects, [104](#), [104](#), [105](#), [108](#), [109](#)
- opal.projects\_databases, [104](#), [105](#), [108](#), [109](#)
- opal.put, [85](#), [97](#), [102](#), [113](#)
- opal.report, [114](#)
- opal.resource, [115](#), [116–126](#)
- opal.resource\_create, [115–118](#), [119](#), [120–126](#)
- opal.resource\_delete, [115–119](#), [120](#), [121–126](#)
- opal.resource\_exists, [115–120](#), [121](#), [122–126](#)
- opal.resource\_extension\_create, [115–121](#), [122](#), [123–126](#)
- opal.resource\_get, [115–122](#), [123](#), [124–126](#)
- opal.resource\_perm, [115–123](#), [124](#), [125](#), [126](#)

- opal.resource\_perm\_add, [115–124](#), [125](#),  
[126](#)
- opal.resource\_perm\_delete, [115–125](#), [126](#)
- opal.resource\_view\_create, [101](#), [127](#), [128](#),  
[138–144](#), [146–148](#), [150–152](#)
- opal.resource\_view\_reconnect, [101](#), [127](#),  
[128](#), [138–144](#), [146–148](#), [150–152](#)
- opal.resources, [115](#), [115](#), [116–126](#)
- opal.resources\_perm, [115](#), [116](#), [116](#),  
[117–126](#)
- opal.resources\_perm\_add, [115](#), [116](#), [117](#),  
[118–126](#)
- opal.resources\_perm\_delete, [115–117](#),  
[118](#), [119–126](#)
- opal.rm, [129](#), [132–134](#)
- opal.sql, [129](#)
- opal.sql\_history, [130](#)
- opal.symbol\_import, [129](#), [132](#), [132](#), [133](#),  
[134](#)
- opal.symbol\_rm, [129](#), [132](#), [133](#), [133](#), [134](#)
- opal.symbol\_save, [129](#), [132](#), [133](#), [134](#)
- opal.symbols, [129](#), [131](#), [133](#), [134](#)
- opal.table, [70](#), [71](#), [79](#), [84](#), [134](#), [135](#), [167–169](#)
- opal.table\_create, [101](#), [127](#), [128](#), [138](#),  
[139–144](#), [146–148](#), [150–152](#)
- opal.table\_delete, [101](#), [127](#), [128](#), [138](#), [139](#),  
[140–144](#), [146–148](#), [150–152](#)
- opal.table\_dictionary\_get, [101](#), [127](#), [128](#),  
[138](#), [139](#), [140](#), [141–144](#), [146–148](#),  
[150–152](#)
- opal.table\_dictionary\_update, [101](#), [127](#),  
[128](#), [138–140](#), [140](#), [142–144](#),  
[146–148](#), [150–152](#)
- opal.table\_exists, [101](#), [127](#), [128](#), [138–141](#),  
[142](#), [143](#), [144](#), [146–148](#), [150–152](#)
- opal.table\_export, [101](#), [127](#), [128](#), [138–142](#),  
[143](#), [144](#), [146–148](#), [150–152](#)
- opal.table\_get, [101](#), [127](#), [128](#), [138–143](#),  
[144](#), [146–148](#), [150–152](#)
- opal.table\_import, [101](#), [127](#), [128](#), [138–144](#),  
[145](#), [146–148](#), [150–152](#)
- opal.table\_perm, [101](#), [127](#), [128](#), [138–144](#),  
[146](#), [146](#), [147](#), [148](#), [150–152](#)
- opal.table\_perm\_add, [101](#), [127](#), [128](#),  
[138–144](#), [146](#), [147](#), [148](#), [150–152](#)
- opal.table\_perm\_delete, [101](#), [127](#), [128](#),  
[138–144](#), [146](#), [147](#), [148](#), [150–152](#)
- opal.table\_save, [101](#), [127](#), [128](#), [138–144](#),  
[146–148](#), [149](#), [150–152](#)
- opal.table\_truncate, [101](#), [127](#), [128](#),  
[138–144](#), [146–150](#), [150](#), [151](#), [152](#)
- opal.table\_view\_create, [101](#), [127](#), [128](#),  
[138–144](#), [146–148](#), [150](#), [151](#), [152](#)
- opal.table\_view\_update, [101](#), [127](#), [128](#),  
[138–144](#), [146–148](#), [150](#), [151](#), [152](#)
- opal.tables, [70](#), [71](#), [79](#), [84](#), [135](#), [135](#),  
[167–169](#)
- opal.tables\_perm, [136](#)
- opal.tables\_perm\_add, [137](#)
- opal.tables\_perm\_delete, [137](#)
- opal.task, [153](#), [154](#), [155](#)
- opal.task\_cancel, [153](#), [154](#), [154](#), [155](#)
- opal.task\_wait, [153–155](#), [155](#)
- opal.tasks, [153](#), [154](#), [155](#)
- opal.taxonomies, [156](#), [157–159](#), [170](#), [171](#)
- opal.taxonomy, [156](#), [156](#), [157–159](#), [170](#), [171](#)
- opal.taxonomy\_delete, [156](#), [157](#), [157](#), [158](#),  
[159](#), [170](#), [171](#)
- opal.taxonomy\_download, [156](#), [157](#), [158](#),  
[159](#), [170](#), [171](#)
- opal.taxonomy\_upload, [156–158](#), [158](#), [159](#),  
[170](#), [171](#)
- opal.terms, [156–159](#), [159](#), [170](#), [171](#)
- opal.token, [160](#), [161–165](#)
- opal.token\_datashield\_create, [160](#), [161](#),  
[161](#), [162–165](#)
- opal.token\_delete, [160–162](#), [162](#), [163–165](#)
- opal.token\_r\_create, [160–163](#), [164](#), [165](#)
- opal.token\_renew, [160–162](#), [163](#), [164](#), [165](#)
- opal.token\_sql\_create, [160–164](#), [165](#)
- opal.tokens, [160](#), [161](#), [162–165](#)
- opal.unload\_package, [86](#), [98](#), [166](#)
- opal.valueset, [70](#), [71](#), [79](#), [84](#), [135](#), [166](#),  
[167–169](#)
- opal.variable, [70](#), [71](#), [79](#), [84](#), [135](#), [167](#), [167](#),  
[168](#), [169](#)
- opal.variable\_summary, [70](#), [71](#), [79](#), [84](#), [135](#),  
[167](#), [168](#), [168](#)
- opal.variables, [70](#), [71](#), [79](#), [84](#), [135](#), [167](#),  
[168](#), [169](#)
- opal.version\_compare, [169](#)
- opal.vocabularies, [156–159](#), [170](#), [171](#)
- opal.vocabulary, [156–159](#), [170](#), [171](#)
- opal.workspace\_restore, [172](#), [172](#), [173](#),  
[174](#)
- opal.workspace\_rm, [172](#), [173](#), [173](#), [174](#)

`opal.workspace_save`, [172](#), [173](#), [174](#)

`opal.workspaces`, [172](#), [173](#), [174](#)