Package: obm (via r-universe)

May 29, 2026

Title Interface to 'OpenBioMaps' Data and Services

Version 2.1

Description Provides access to selected functions and data available through any 'OpenBioMaps' <https://openbiomaps.org> server instance. 'OpenBioMaps' is an open-source biodiversity data management platform designed for conservation professionals and researchers. User authentication and access control are handled through 'OpenBioMaps' login credentials.

Depends R (>= 3.1.0)

License GPL-3

Imports httr, rjson, jsonlite, RPostgreSQL, DBI, utils

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Miklós Bán [aut, cre]

Maintainer Miklós Bán <banm@vocs.unideb.hu>

Config/pak/sysreqs libssl-dev

Repository https://miklosban.r-universe.dev

Date/Publication 2026-05-15 23:16:02 UTC

RemoteUrl https://github.com/cran/obm

RemoteRef HEAD

RemoteSha 544c990e91a3e37849febc0df2c89121ab277a5d

Contents

`obm_auth` *Log in to an OpenBioMaps server*

Description

This function allows you to connect to and OBM server.

Usage

```
obm_auth(username = "", password = "", verbose = FALSE, paranoid = TRUE)
```

Arguments

`username`	Your OBM username (email)
`password`	Your password
`verbose`	print verbose feedback messages - default is FALSE
`paranoid`	hide password while typing (on Linux) - default is TRUE

Value

true if successful and false if not

Examples

```
# default interactive authentication
obm_auth()

# authentication with username and password
obm_auth('foo@google.com','abc123')
```

obm_computation *Background Computation Function*

Description

This experimental function allows to manage background computation on an OpenBioMaps server.

Usage

```
obm_computation(
  action = "",
  data_files = NULL,
  params = NULL,
  config_file = NULL
)
```

Arguments

action	post, get-results, get-status
data_files	list of files to sending
params	computation control params
config_file	is The computation_conf.yml file

Value

data frame or boolean

Examples

```
# Manage computational packs
results <- obm_computation(action='post',
                           data_files='examples/scripts/bombina.R',
                           params=list(package = 'my_project'),
                           config_file='examples/computation_conf.yml')
```

obm_connect *Connect to a server with a shared link*

Description

This function allows to connect to an OBM server with a shared link It using client_credentials authentication, so it is returning an access_token Return an oauth token

Usage

```
obm_connect(link = "", verbose = FALSE)
```

Arguments

link	an url link
verbose	print verbose feedback messages - default is FALSE

Value

boolean

Examples

```
# Interactive connect
obm_connect()
# Non interactive connect
token <- obm_connect('abcdefghikl123456789')
```

---

obm_get *Get Information/Data from an OpenBioMaps project*

Description

This function get data or information from a project. Supported functions (APIv3): get_data: Get records from an SQL table or view, get_public_data: Get publicly available records from an SQL table or view, get_tables: Get list of SQL tables (or column list) and VIEWS in a project, get_form_list, get_form_data: Get upload form information For API v3 (api_version >= 3.0), 'get_data' uses GraphQL for powerful filtering and data selection. See the [GraphQL User Guide](https://gitlab.com/openbior project-api/-/blob/main/GraphQLUserGuide.md) for more details on query possibilities.

Usage

```
obm_get(
  scope = "",
  control_condition = NULL,
  condition = NULL,
  table = get_obm("project")
)
```

Arguments

scope	A supported obm scope, e.g. get_data, get_form_list, get_tables
control_condition	Control condition. - In API v3: 'limit=LIMIT:OFFSET' format - In API v2: SQL-like, e.g., 'limit=10:1'

condition	list - A condition based on column in your table. - In API v3 (GraphQL): Can include operators like ‘iequals‘, ‘AND‘, ‘OR‘, and special keys: - ‘fields‘: vector of columns to return - ‘schema‘: database schema (defaults to ’public’) - ‘table‘: database table (defaults to project name) - In API v2: Mostly key-value pairs, e.g., list(species = ’Parus palustris’)
table	optional table from project (fallback if not in condition list)

Details

The following code *illustrates* how you could use this function in practice, but it is **not meant to be run directly** (e.g., depends on external data).

```
# — API v3 GraphQL Examples —

# get rows where column 'species' is 'Parus palustris' (case-insensitive)
data <- obm_get('get_data', condition=list(species = list(iequals = 'Parus palustris')))

# get specific fields from a specific table and schema
data <- obm_get('get_data',
                condition=list(
                  schema = "public",
                  table = "dead_animals",
                  fields = c("obm_id", "faj", "hely"),
                  faj = list(iequals = "asio otus")
                ))

# get all fields from the default table
data <- obm_get('get_data',
                condition=list(
                  fields = '*',
                  faj = list(iequals = "asio otus")
                ))

# get 1000 rows with offset 0 - using control_condition
data <- obm_get('get_data', 'limit=1000:0')

# get list of available forms
data <- obm_get('get_form_list')

# get list of available tables in the project
obm_get('get_tables')

# get table details (columns) in v3
obm_get('get_tables', condition=list(schema="public", table="dead_animals"))

# — API v2 Examples —

results <- obm_get('get_form_list')
```

```
form <- obm_get('get_form_data',results[,]$published_form_id)

# get data rows from the main table from 39980 to 39988
data <- obm_get('get_data',condition=list(obm_id = '39980:39988'))

#get rows from the main table where column 'species' is 'Parus palustris'
data <- obm_get('get_data',condition=list(species = 'Parus palustris'))

#get 100 rows only from filtered query
data <- obm_get('get_data','limit=100:0',condition=list(species = 'Parus palustris'))

#get all data from the default/main table
data <- obm_get('get_data','*')

#get data from a non-default table
data <- obm_get('get_data','*',table='additional_data')
```

Value

a data.frame, list or error message

Examples

```
data <- obm_get('get_data', condition=list(species = list(iequals = 'Parus palustris')))
```

---

obm_init *Initialise connection to an OpenBioMaps server*

Description

This function is initiating an OBM connection.

Usage

```
obm_init(
  project = "",
  url = "openbiomaps.org",
  scope = c(),
  verbose = FALSE,
  api_version = 3
)
```

Arguments

project	Which project you would like to connect? It has a short name.
url	project server's web url DEFAULT is https://openbiomaps.org
scope	vector of required scopes. DEFAULT is ok usually
verbose	print verbose feedback messages - default is FALSE
api_version	API version. 3.0 or higher enables the new REST and GraphQL API.

Value

boolean

Examples

```
# connect with API v3 to the dead_animals project on the default openbiomaps.org server
obm_init(project='dead_animals')
# connect to a database on the default server (openbiomaps.org) with API v2.5
obm_init(project='dead_animals', api_version=2.5)
# connect on the local server instance to the butterfly database project
obm_init(url='http://localhost/biomaps',project='butterflies')
```

---

obm_put *Put data unsing obm forms*

Description

This function allows put data into an OpenBioMaps server. For API v3 (api_version >= 3.0), 'put_data' is supported. Note: In API v3, 'put_data' currently supports single observation upload only.

Usage

```
obm_put(
  scope = NULL,
  form_header = NULL,
  data_file = NULL,
  media_file = NULL,
  form_id = "",
  form_data = "",
  soft_error = "",
  data_table = NULL
)
```

Arguments

scope	currently put_data supported.
form_header	database column names vector, if missing default is the full list from the form
data_file	a csv file with header row (API v2)
media_file	a media file to attach
form_id	the form's id
form_data	JSON array or data.frame of data. In v3, only the first row of a data.frame is uploaded.
soft_error	JSON array of 'Yes' strings (or translations of it) to skip soft error messages (API v2)
data_table	a database table name

Details

The following code *illustrates* how you could use this function in practice, but it is **not meant to be run directly** (e.g., depends on external data).

```
# Using own list of columns
results <- obm_get('get_form_list')
form_id <- results$form_id[1] # e.g. 57

response <- obm_put('put_data',columns[1:3],form_id=form_id,data_file='examples/test_upload.csv')

# Using default columns list:
response <- obm_put(scope='put_data',form_id=57,csv_file='examples/test_upload.csv')

# JSON upload
data <- matrix(c(
                 c("Tringa totanus",'egyed',"AWBO",'10','POINT(47.1 21.3)'),
                 c("Tringa flavipes",'egyed',"BYWO",'2','POINT(47.3 21.4)')
               ),ncol=5,nrow=2,byrow=TRUE)
colnames(data)<-c("species","numerosity","location","quantity","geometry")

response <- obm_put(
               scope='put_data',
               form_id=57,
               form_data=as.data.frame(data),
               form_header=c('faj','szamossag','hely','egyedszam'))

# With attached files
data <- matrix(c(c("Tringa totanus",'egyed',"AWBO",'10','numbers.odt'),
                 c("Tringa flavipes",'egyed',"BYWO",'2','observation.jpg')
               ),ncol=5,nrow=2,byrow=TRUE)
colnames(data)<-c("species","numerosity","location","quantity",'Attach')

response <- obm_put(
```

```
                scope='put_data',
                form_id=57,
                form_data=as.data.frame(data),
           form_header=c('species','numerosity','location','quantity','obm_files_id'),
                media_file=c('examples/numbers.odt','examples/observation.jpg'))
```

Value

json or boolean

Examples

```
csv_path <- system.file("examples/test_upload.csv", package = "obm")
csv_data <- read.csv(csv_path)
columns <- names(csv_data)
response <- obm_put('put_data',columns[1:3],form_id=57,data_file='examples/test_upload.csv')
```

`obm_refresh_token` *Refresh an expired OBM access token*

Description

This function requests a new access token using the stored refresh token. It should normally be called internally by other OBM functions when the current token has expired.

Usage

```
obm_refresh_token(
  token = NULL,
  url = get_obm("token_url"),
  client_id = get_obm("client_id", "R"),
  verbose = FALSE
)
```

Arguments

`token`	Optional explicit refresh token (defaults to stored one)
`url`	OAuth2 token endpoint URL (from obm_init())
`client_id`	OAuth2 client identifier ("R" by default)
`verbose`	Print HTTP and token details

Value

Logical TRUE if token refreshed successfully, FALSE otherwise

Examples

```
obm_refresh_token(token='ABCD1234')
```

---

obm_repo *Repozitorium manager function*

Description

This experimental function allows manage Dataverse repozitorium and datasets.

Usage

```
obm_repo(scope = NULL, data_table = NULL, params = NULL)
```

Arguments

scope	get or put
data_table	OBM project table
params	list which contains parameters for repozitorium

Details

The following code *illustrates* how you could use this function in practice, but it is **not meant to be run directly** (e.g., depends on external data).

```
# Set the default server/project-repo for each of the following operations
#   - default is 0
#   - get possible names from server_conf query above
obm_repo('set', params=list(REPO='ABC-REPO'))
obm_repo('set', params=list(REPO='ABC-REPO', PARENT='dataverse/parent'))

# Listing dataverse
results <- obm_repo('get', params=list(type='dataverse', contents=1))
results <- obm_repo('get', params=list(type='dataverse'))

# Getting content of the named dataverse
results <- obm_repo('get', params=list(id='MY-DATAVERSE'))

# Get JSON Representation of a Dataset
results <- obm_repo('get', params=list(type='datasets',
                                persistentUrl='https://doi.org/xxx/xxx/xxx'))
results <- obm_repo('get', params=list(type='datasets', id='XABC-1'))

# Get versions of dataset
results <- obm_repo('get',params=list(type='datasets', id=42, version=''))
```

```
results <- obm_repo('get',params=list(type='datasets', id=42, version=':draft'))

# Get files of dataset
results <- obm_repo('get',params=list(type='datasets', id=42, files='', version=''))

# Get a file
results<-obm_repo('get',params=list(type='datafile', id=83))
results<-obm_repo('get',params=list(type='datafile', id=83, version=':draft'))

# Create a dataverse
results <- obm_repo('put',params=list(type='dataverse'))

# Create a dateset
results <- obm_repo('put',params=list(type='datasets', dataverse='my_dataset'))

# Add file to dataset (referenced by id or persistentUrl)
results <- obm_repo('put',params=list(type='datafile',
                                      file='examples/test_upload.csv',
                                      id='XABC-1' ))
results <- obm_repo('put',params=list(type='datafile',
                                      file='examples/test_upload.csv',
                                   persistentUrl='https://doi.org/xxx/xxx/xxx'))

# Add object as file to dataset (referenced by id or persistentUrl)
# - automatically convert data object to JSON
# - returning with the last file's state
init.df <- data.frame()
results <- obm_repo('put',params=list(type='datafile',
                                      id='XABC-1',
                     data=list(results=res.list,
                               init_params=init.df)))

# Delete file
results <- obm_repo('delete',params=list(type='datafile',
                                         id='XABC-1',
                                         PARENT_DATAVERSE='dataverse/parent'))

# Set settings
# obm_repo('set',params=list(type='dataset',id='XABC-1'))
```

Value

data frame or boolean

Examples

```
# Getting server conf
results <- obm_repo('get', params=list(server_conf=1))
```

obm_set *Set Function*

Description

Experimental function in APIv2! This function allows you to set rules for obm_get.

Usage

```
obm_set(scope = "", condition = "")
```

Arguments

scope	Which scope? e.g. set_join
condition	A text condition based on column in your table

Value

list, json or boolean

Examples

```
# automatically join tables
data <- obm_set('set_join',c('dead_animals','dead_animals_history'))
```

obm_sql_query *SQL Interface*

Description

It is a simple SQL Query interface function

Usage

```
obm_sql_query(
  sqlcmd,
  username = "",
  password = "",
  paranoid = TRUE,
  port = 5432,
  database = "gisdata"
)
```

Arguments

sqlcmd	a valid SQL command
username	most probably automatically set by create_pg_user module
password	most probably automatically set by create_pg_user module
paranoid	password prompt type
port	postgres server port, default is 5432
database	remote database name, default is gisdata

Value

sql data object or boolean

Examples

```
obm_sql_query("SELECT DATE_PART('day', enddate::timestamp - startdate::timestamp) AS days
               FROM nestboxes WHERE enddate IS NOT NULL AND startdate IS NOT NULL
               ORDER BY days")
```

Index