Package: deposits (via r-universe)

July 12, 2024

Title A universal client for depositing and accessing research data anywhere

Version 0.2.1.057

Description A universal client for depositing and accessing research data anywhere. Currently supported services are zenodo and figshare.

License MIT + file LICENSE

URL https://docs.ropensci.org/deposits/

BugReports https://github.com/ropenscilabs/deposits/issues

Imports checkmate, fs, here, httr2, jsonlite, jsonvalidate, methods, R6, withr, xml2

Suggests frictionless, httptest2, knitr, pkgbuild, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/ropenscilabs/deposits

RemoteRef main

RemoteSha aaaf7f7ed3a3f6a587792032bf5ba245b5e3631b

Contents

`dcmi_terms` *Get names of DCMI terms*

Description

The Dublin Core Metadata Initiative defines a set of terms at https://www.dublincore.org/specifications/dublin-core/dcmi-terms/.

Usage

```
dcmi_terms(term = NULL)
```

Arguments

`term`	If specified, match term to official DCMI nomenclature, and return single match.

Value

A character vector of DCMI terms.

See Also

Other meta: `deposits_metadata_template()`, `figshare_categories()`

`depositsClient` *depositsClient*

Description

An R6 client for managing deposits on external services, currently including Figshare and Zenodo. Use of a 'deposits' client is controlled by the methods listed below. Those looking for help with client usage are advised to head to that section.

Value

A `depositsClient` class (R6 class)

Public fields

`service` (character) of deposits host service.

`sandbox` (logical) Connect client with sandbox if `TRUE` (zenodo only)

`deposits` (data.frame) Current deposits hosted on service, one row per deposit.

`frictionless` (logical) Default behaviour of `TRUE` assumes uploads are data files in rectangular form, able to be described by **frictionless** metadata. **frictionless** integration is by-passed when this parameter if `FALSE`.

`url_base` (character) Base URL of host service API

`url_service` (character) URL of deposit service

`id` (integer) Deposit identifier from host service.

`headers` (list) list of named headers

`hostdata` (list) Data as stored by host platform

`metadata` holds list of DCMI-compliant metadata.

`local_path` holds path to local directory (not file) containing current deposit.

Methods

Public methods:

- `depositsClient$new()`
- `depositsClient$print()`
- `depositsClient$deposit_add_resource()`
- `depositsClient$deposit_delete()`
- `depositsClient$deposit_delete_file()`
- `depositsClient$deposit_download_file()`
- `depositsClient$deposit_embargo()`
- `depositsClient$deposit_fill_metadata()`
- `depositsClient$deposit_new()`
- `depositsClient$deposit_prereserve_doi()`
- `depositsClient$deposit_publish()`
- `depositsClient$deposit_retrieve()`
- `depositsClient$deposit_service()`
- `depositsClient$deposit_update()`
- `depositsClient$deposit_upload_file()`
- `depositsClient$deposit_version()`
- `depositsClient$deposits_list()`
- `depositsClient$deposits_methods()`
- `depositsClient$deposits_search()`

Method `new()`**:** Create a new `depositsClient` object, as an **R6** client with methods listed via `deposits_emthods()`.

Usage:

```
depositsClient$new(service, metadata = NULL, sandbox = FALSE, headers = NULL)
```

Arguments:

`service` (character) Name of a deposits service (see deposits_services).

`metadata` Either of one two possible ways of defining metadata:

- The name (or full path) or a local file containing metadata constructed with deposits_metadata_template;
- A names list of metadata with names matching values given by dcmi_terms, and values specified as individual character strings or lists for multiple entries.

`sandbox` If `TRUE`, connect client to sandbox, rather than actual API endpoint (for "zenodo" only).

`headers` Any acceptable headers. See examples in **httr2** package.

Returns: A new `depositsClient` object

Examples:

```
\dontrun{
cli <- depositsClient$new (service = "zenodo", sandbox = TRUE)
# List methods of client:
cli$deposits_methods ()
# List all current deposits associated with user token:
cli$deposits_list ()

# Once a deposit has locally-stored metadata associated with it, only
# that parameter is needed.
path <- tempfile (pattern = "data") # A directory for data storage
dir.create (path)
f <- file.path (path, "beaver1.csv")
write.csv (datasets::beaver1, f, row.names = FALSE)
metadata <- list (
    creator = list (list (name = "P. S. Reynolds")),
    created = list (publisherPublication = "1994-01-01"),
    title = "Time-series analyses of beaver body temperatures",
    description = "Original source of 'beaver' dataset."
)
cli <- depositsClient$new (service = "figshare", metadata = metadata)
cli$deposit_new ()
cli$deposit_upload_file (f)

# A new deposits client may then be constructed by passing the data
# directory as the 'metadata' parameter:
cli <- depositsClient$new (metadata = path)
}
```

Method `print()`**:** `print` method for the `depositsClient` class, providing an on-screen overview of current contents and structure of client.

Usage:

```
depositsClient$print(x, ...)
```

Arguments:

`x` self

`...` ignored

Method `deposit_add_resource()`**:** Generate a local "datapackage.json" file, and/or add metadata from client. A "resource" must be readable by the **frictionless** package, generally meaning either a 'datapackage.json' file, or a rectangular structure able to be read and represented as a `data.frame`. See https://docs.ropensci.org/frictionless/ for details.

Usage:

```
depositsClient$deposit_add_resource(path)
```

Arguments:

`path` Path to local resource to be added to client. May name an individual file or a directory.

Returns: (Invisibly) Updated 'deposits' client

Method `deposit_delete()`**:** Deleted a specified deposit from the remote service. This removes the deposits from the associated service, along with all corresponding 'hostdata' in the local client.

Usage:

```
depositsClient$deposit_delete(deposit_id = NULL)
```

Arguments:

`deposit_id` Integer identifier of deposit (generally obtained from `list_deposits` method).

Returns: (Invisibly) Updated 'deposits' client

Method `deposit_delete_file()`**:** Delete a single from a deposits service.
This does not modify the "datapackage.json" file, either locally or on a service.

Usage:

```
depositsClient$deposit_delete_file(filename)
```

Arguments:

`filename` Name of file to be deleted as recorded on service.

`deposit_id` The 'id' number of deposit from which file is to be deleted. If not specified, the 'id' value of current deposits client is used.

Returns: (Invisibly) Updated 'deposits' client

Examples:

```
\dontrun{
# Initiate deposit and fill with metadata:
metadata <- list (
    title = "Time-series analyses of beaver body temperatures",
    description = "Original source of 'beaver2' data",
    creator = list (list (name = "P.S. Reynolds")),
    created = "1994-01-01T00:00:00",
    publisher = "Case Studies in Biometry"
)
cli <- depositsClient$new (
    service = "zenodo",
    sandbox = TRUE,
    metadata = metadata
)
cli$deposit_new ()
```

```
# Create some local data and upload to deposit:
path <- fs::path (fs::path_temp (), "beaver.csv")
write.csv (datasets::beaver2, path)
cli$deposit_upload_file (path = path)

# Confirm that uploaded files include \pkg{frictionless}
# "datapackage.json" file, and also that local version has been
# created:
cli$hostdata$files

# Then delete one of those files:
cli$deposit_delete_file ("datapackage.json")
}
```

Method `deposit_download_file()`**:** Download a specified 'filename' from a deposit.

Usage:

```
depositsClient$deposit_download_file(
  filename,
  deposit_id = NULL,
  path = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments:

`filename` The name of the file to be download as specified in the deposit.

`deposit_id` The 'id' number of deposit which file is to be downloaded from. If not specified, the 'id' value of current deposits client is used.

`path` The local directory where file is to be downloaded.

`overwrite` Do not overwrite existing files unless set to `TRUE`.

`quiet` If `FALSE`, display download progress.

Returns: The full path of the downloaded file.

Method `deposit_embargo()`**:** Embargo a deposit prior to publication.

Usage:

```
depositsClient$deposit_embargo(
  embargo_date = NULL,
  embargo_type = c("deposit", "file"),
  embargo_reason = NULL
)
```

Arguments:

`embargo_date` Date of expiry of embargo. If the `deposit_publish()` method has been called, deposit will automatically be published after this date, and will not be published, nor publicly accessible, prior to this date.

`embargo_type` For Figshare service only, which allows embargoes for entire deposits or single files. Ignored for other services.

`embargo_reason` For Figshare service only, an optional text string describing reasons for embargo.

Returns: (Invisibly) Updated deposits client with additional embargo information.

Method `deposit_fill_metadata()`**:** Fill deposits client with metadata.

Usage:

```
depositsClient$deposit_fill_metadata(metadata = NULL)
```

Arguments:

`metadata` Either one of two possible ways of defining metadata:

- The name (or full path) or a local file containing metadata constructed with deposits_metadata_template;
- A names list of metadata with names matching values given by dcmi_terms, and values specified as individual character strings or lists for multiple entries.

Returns: (Invisibly) Updated deposits client with metadata inserted.

Method `deposit_new()`**:** Initiate a new deposit on the external deposits service.

Usage:

```
depositsClient$deposit_new(prereserve_doi = TRUE, quiet = FALSE)
```

Arguments:

`prereserve_doi` If `TRUE`, a Digital Object Identifier (DOI) is prereserved on the nominated service, and returned in the "hostdata". This DOI will also be inserted in the "identifier" field of the client metadata.

`quiet` If `FALSE` (default), print integer identifier of newly created deposit.

Returns: (Invisibly) Updated deposits client which includes data on new deposit

Method `deposit_prereserve_doi()`**:** Prereserve a DOI. This is generally done when a deposit is first initialised, via the `prereserve_doi` parameter. This method exists only to subsequently prereserve a DOI for deposits initiated with `prereserve_doi = FALSE`.

Usage:

```
depositsClient$deposit_prereserve_doi()
```

Returns: (Invisibly) Updated 'deposits' client

Method `deposit_publish()`**:** Publish a deposit. This is an irreversible action which should only be called if you are really sure that you want to publish the deposit. Some aspects of published deposits can be subsequently edited, but they can never be deleted.

Usage:

```
depositsClient$deposit_publish()
```

Returns: (Invisibly) Updated 'deposits' client

Method `deposit_retrieve()`**:** Retrieve a specified deposit and store information in local client.

Usage:

```
depositsClient$deposit_retrieve(deposit_id, quiet = FALSE)
```

Arguments:

`deposit_id` The 'id' number of deposit for which information is to be retrieved.

`quiet` If `FALSE` (default), display information on screen on any issues encountered in retrieving deposit.

Returns: (Invisibly) Updated 'deposits' client

Method `deposit_service()`**:** Switch external services associated with a `depositsClient` object.

Usage:

```
depositsClient$deposit_service(service = NULL, sandbox = FALSE, headers = NULL)
```

Arguments:

`service` (character) Name of a deposits service (see deposits_services).

`sandbox` If `TRUE`, connect client to sandbox, rather than actual API endpoint (for "zenodo" only).

`headers` Any acceptable headers. See examples in **httr2** package.

Returns: (Invisibly) Updated deposits client.

Method `deposit_update()`**:** Update a remote (online) deposit with local metadata.

Usage:

```
depositsClient$deposit_update(deposit_id = NULL, path = NULL)
```

Arguments:

`deposit_id` (Optional) The 'id' number of deposit to update. If not specified, the 'id' value of current deposits client is used.

`path` (Optional) If given as path to single file, update that file on remote service. If given as a directory, update all files within that directory on remote service. If not given, path will be taken from client's "local_path" field. Only files for which local versions have been changed will be uploaded.

Returns: (Invisibly) Updated deposits client.

Method `deposit_upload_file()`**:** Upload a local file or folder to an specified deposit, or update an existing version of file with new local version.

Usage:

```
depositsClient$deposit_upload_file(
  path = NULL,
  deposit_id = NULL,
  overwrite = FALSE,
  compress = c("no", "zip", "tar"),
  quiet = FALSE
)
```

Arguments:

`path` Either single file name or full path to local file or folder to be uploaded. If a single file name, the path if taken from the client's "local_path" field. If the file to be uploaded is able to be read as a tabular data file, an associated **frictionless** "datapackage.json" file will also be uploaded if it exists, or created if it does not. The metadata within a client will also be used to fill or update any metadata within the "datapackage.json" file.

`deposit_id` The 'id' number of deposit which file is to be uploaded to. If not specified, the 'id' value of current deposits client is used.

`overwrite` Set to `TRUE` to update existing files by overwriting.

`compress` One of "no" (default), "zip", or "tar", where the latter two will compress data in the chosen binary format prior to uploading. All files are individually compressed; uploading binary archives of multiple files is not recommended, as it prevents people downloading selections of those files.

`quiet` If `FALSE` (default), display diagnostic information on screen.

Returns: (Invisibly) Updated 'deposits' client

Examples:

```
\dontrun{
# Initiate deposit and fill with metadata:
metadata <- list (
    title = "Time-series analyses of beaver body temperatures",
    description = "Original source of 'beaver2' data",
    creator = list (list (name = "P.S. Reynolds")),
    created = "1994-01-01T00:00:00",
    publisher = "Case Studies in Biometry"
)
cli <- depositsClient$new (
    service = "zenodo",
    sandbox = TRUE,
    metadata = metadata
)
cli$deposit_new ()

# Create some local data and upload to deposit:
path <- fs::path (fs::path_temp (), "beaver.csv")
write.csv (datasets::beaver2, path)
cli$deposit_upload_file (path = path)

# Confirm that uploaded files include \pkg{frictionless}
# "datapackage.json" file, and also that local version has been
# created:
cli$hostdata$files
fs::dir_ls (fs::path_temp (), regexp = "datapackage")
}
```

Method `deposit_version()`**:** Start a new version of a published deposit, based on current client metadata. This method is not available for Figshare.

Usage:

```
depositsClient$deposit_version()
```

Returns: (Invisibly) Updated 'deposits' client

Method `deposits_list()`**:** Update 'deposits' item of current deposits for given service. The list of deposits contained within the "deposits" item of a client may not be up-to-date; this method can be used for force synchronisation with the external service, so that "deposits" lists all current deposits.

Usage:

```
depositsClient$deposits_list()
```

Returns: (Invisibly) Updated 'deposits' client

Examples:

```
\dontrun{
cli <- depositsClient$new (service = "zenodo", sandbox = TRUE)
print (cli)
# ... then if "Current deposits" does not seem up-to-date:
cli$deposits_list ()
# That will ensure that all external deposits are then listed,
# and can be viewed with:
cli$deposits
}
```

Method `deposits_methods()`**:** List public methods of a 'deposits' client.

Usage:

```
depositsClient$deposits_methods()
```

Returns: Nothing; methods are listed on screen.

Method `deposits_search()`**:** Search all public deposits.

Usage:

```
depositsClient$deposits_search(
  search_string = NULL,
  page_size = 10L,
  page_number = 1L,
  ...
)
```

Arguments:

`search_string` Single string to search for

`page_size` Number of records to return in one page

`page_number` Starting page for return results; used in combination with 'page_size' for pagination.

`...` Named pairs of query parameters. Zenodo parameters are described at https://developers.zenodo.org/#list36, and currently include:

- status: either "draft" or "published"
- sort: either "bestmatch" (the default) or "mostrecent"
- all_versions: Either "true" or "false"
- communities: Search for deposits only within specified communities
- type: Return deposits only of specified type
- subtype: Return deposits only of specified subtype
- bound: A geolocation bounding box
- custom: Custom keywords

Figshare parameters are described at https://docs.figshare.com/#articles_search, and currently include:

- resource_doi: Only return deposits matching this 'resource_doi'

- item_type: Return deopsits of specified type (as integer).
- doi: Only return deposits matching this DOI
- handle: Only return deposits matching this handle
- project_id: Only return deposits from within specified project
- order: Order for sorting results; one of "published_date", "modified_date", "views", "shares", "downloads", or "cites"
- search_for: Search term.
- order_direction: "asc" or "desc"
- institution: Only return deposits from specified institution (as integer)
- group: Only return deposits from specified group (as integer)
- published_since: Only return deposits published since specified date (as YYYY-MM-DD)
- modified_since: Only return deposits modified since specified date (as YYYY-MM-DD)

Returns: A `data.frame` of data on deposits matching search parameters (with format depending on the deposits service.)

Examples:

```
\dontrun{
cli <- depositsClient$new (service = "figshare")
search_results <- cli$deposits_search (
    search_string = "Text string query",
    page_size = 5L
)
# The 'search_string' can be used to specify precise searches:
cli <- depositsClient$new (service = "zenodo")
search_results <-
   cli$deposits_search ("keywords='frictionlessdata'&type='dataset'")
}
```

Note

This method is generally intended to be used for private deposits; that is, to edit deposits prior to publication. It is nevertheless possible to edit published deposits on Zenodo, and this method will do so if called on a public Zenodo deposit. The updated data and/or metadata will not be publicly visible until the deposit is again published with the `deposit_publish()` method.

Examples

```
## Not run:
# make a client
cli <- depositsClient$new ("zenodo") # or:
cli <- depositsClient$new ("figshare")
print (cli)

# methods
cli$deposits_list ()

# Fill depositsClient metadata
```

```
metadata <- list (
    title = "New Title",
    abstract = "This is the abstract",
    creator = list (list (name = "A. Person"), list (name = "B. Person"))
)
cli$deposit_fill_metadata (metadata)
print (cli)

# or pass metadata directly at construction of new client
cli <- depositsClient$new ("figshare", metadata = metadata)

## End(Not run)

## ------------------------------------------------
## Method `depositsClient$new`
## ------------------------------------------------

## Not run:
cli <- depositsClient$new (service = "zenodo", sandbox = TRUE)
# List methods of client:
cli$deposits_methods ()
# List all current deposits associated with user token:
cli$deposits_list ()

# Once a deposit has locally-stored metadata associated with it, only
# that parameter is needed.
path <- tempfile (pattern = "data") # A directory for data storage
dir.create (path)
f <- file.path (path, "beaver1.csv")
write.csv (datasets::beaver1, f, row.names = FALSE)
metadata <- list (
    creator = list (list (name = "P. S. Reynolds")),
    created = list (publisherPublication = "1994-01-01"),
    title = "Time-series analyses of beaver body temperatures",
    description = "Original source of 'beaver' dataset."
)
cli <- depositsClient$new (service = "figshare", metadata = metadata)
cli$deposit_new ()
cli$deposit_upload_file (f)

# A new deposits client may then be constructed by passing the data
# directory as the 'metadata' parameter:
cli <- depositsClient$new (metadata = path)

## End(Not run)

## ------------------------------------------------
## Method `depositsClient$deposit_delete_file`
## ------------------------------------------------

## Not run:
# Initiate deposit and fill with metadata:
metadata <- list (
```

```
    title = "Time-series analyses of beaver body temperatures",
    description = "Original source of 'beaver2' data",
    creator = list (list (name = "P.S. Reynolds")),
    created = "1994-01-01T00:00:00",
    publisher = "Case Studies in Biometry"
)
cli <- depositsClient$new (
    service = "zenodo",
    sandbox = TRUE,
    metadata = metadata
)
cli$deposit_new ()

# Create some local data and upload to deposit:
path <- fs::path (fs::path_temp (), "beaver.csv")
write.csv (datasets::beaver2, path)
cli$deposit_upload_file (path = path)

# Confirm that uploaded files include \pkg{frictionless}
# "datapackage.json" file, and also that local version has been
# created:
cli$hostdata$files

# Then delete one of those files:
cli$deposit_delete_file ("datapackage.json")

## End(Not run)

## ------------------------------------------------
## Method `depositsClient$deposit_upload_file`
## ------------------------------------------------

## Not run:
# Initiate deposit and fill with metadata:
metadata <- list (
    title = "Time-series analyses of beaver body temperatures",
    description = "Original source of 'beaver2' data",
    creator = list (list (name = "P.S. Reynolds")),
    created = "1994-01-01T00:00:00",
    publisher = "Case Studies in Biometry"
)
cli <- depositsClient$new (
    service = "zenodo",
    sandbox = TRUE,
    metadata = metadata
)
cli$deposit_new ()

# Create some local data and upload to deposit:
path <- fs::path (fs::path_temp (), "beaver.csv")
write.csv (datasets::beaver2, path)
cli$deposit_upload_file (path = path)
```

```
# Confirm that uploaded files include \pkg{frictionless}
# "datapackage.json" file, and also that local version has been
# created:
cli$hostdata$files
fs::dir_ls (fs::path_temp (), regexp = "datapackage")

## End(Not run)

## ------------------------------------------------
## Method `depositsClient$deposits_list`
## ------------------------------------------------

## Not run:
cli <- depositsClient$new (service = "zenodo", sandbox = TRUE)
print (cli)
# ... then if "Current deposits" does not seem up-to-date:
cli$deposits_list ()
# That will ensure that all external deposits are then listed,
# and can be viewed with:
cli$deposits

## End(Not run)

## ------------------------------------------------
## Method `depositsClient$deposits_search`
## ------------------------------------------------

## Not run:
cli <- depositsClient$new (service = "figshare")
search_results <- cli$deposits_search (
    search_string = "Text string query",
    page_size = 5L
)
# The 'search_string' can be used to specify precise searches:
cli <- depositsClient$new (service = "zenodo")
search_results <-
   cli$deposits_search ("keywords='frictionlessdata'&type='dataset'")

## End(Not run)
```

`deposits_metadata_template` *Write an empty metadata template to local file*

Description

The fields are those defined by the Dublin Core Metadata Initiative (DCMI), defined at https://www.dublincore.org/specifications/dublin-core/dcmi-terms/. The template produced by this function is in `json` format which can be manually edited to provide metadata for a deposit.

Usage

```
deposits_metadata_template(filename = NULL)
```

Arguments

filename	Name or full path to local file where template is to be written. This file will be created. If a file of that name already exists, it must first be deleted. The file extension '.json' will be automatically appended.

Value

(Invisibly) TRUE if local file successfully created; otherwise FALSE.

See Also

Other meta: dcmi_terms(), figshare_categories()

Examples

```
filename <- tempfile (fileext = ".json")
deposits_metadata_template (filename)
# then edit that file to complete metadata
```

---

deposits_services *List all deposits services and associated URLs*

Description

List all deposits services and associated URLs

Usage

```
deposits_services()
```

Value

A data.frame with name and url values for each accessible service.

Examples

```
s <- deposits_services ()
```

figshare_categories *Select figshare categories and return corresponding integer identifier.*

Description

These identifiers should then be added in deposit metadata as, for example, `subject(categories=c(1,2))`.

Usage

```
figshare_categories()
```

See Also

Other meta: `dcmi_terms()`, `deposits_metadata_template()`

get_deposits_token *Retrieve a token for a specified deposits service.*

Description

Tokens should be stored as local environment variables, optionally defined in a `~/.Renviron` file, and should contain the name of the desired deposits service.

Usage

```
get_deposits_token(service = NULL, sandbox = FALSE)
```

Arguments

`service`	Name of desired service; must be a value in the "name" column of deposits_services.
`sandbox`	If `TRUE`, retrieve token for sandbox, rather than actual API.

Value

API token for nominated service.

Examples

```
## Not run:
token <- get_deposits_token (service = "figshare")

## End(Not run)
```

Index