Package 'zen4R'

September 15, 2021

Version 0.5

Date 2021-09-14

Title Interface to 'Zenodo' REST API

Maintainer Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Depends R (>= 3.3.0), methods

Imports R6, httr, jsonlite, xml2, keyring, tools

Suggests testthat, parallel

Description Provides an Interface to 'Zenodo' (<https://zenodo.org>) REST API, including management of depositions, attribution of DOIs by 'Zenodo' and upload and download of files.

License MIT + file LICENSE

URL https://github.com/eblondel/zen4R

BugReports https://github.com/eblondel/zen4R/issues

LazyLoad yes

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Author Emmanuel Blondel [aut, cre] (<https://orcid.org/0000-0002-5870-5762>), Julien Barde [ctb] (<https://orcid.org/0000-0002-3519-6141>), Stephen Eglen [ctb] (<https://orcid.org/0000-0001-8607-8025>), Hans Van Calster [ctb] (<https://orcid.org/0000-0001-8595-8426>), Floris Vanderhaeghe [ctb] (<https://orcid.org/0000-0002-6378-6229>)

Repository CRAN

Date/Publication 2021-09-15 10:50:02 UTC

R topics documented:

`download_zenodo` *download_zenodo*

Description

`download_zenodo` allows to download archives attached to a Zenodo record, identified by its DOI or concept DOI.

Usage

```
download_zenodo(
  doi,
  path = ".",
  files = list(),
  logger = NULL,
  quiet = FALSE,
  ...
)
```

Arguments

`doi`	a Zenodo DOI or concept DOI
`path`	the target directory where to download files
`files`	subset of filenames to restrain to download. If ignored, all files will be downloaded.
`logger`	a logger to print Zenodo API-related messages. The logger can be either NULL, "INFO" (with minimum logs), or "DEBUG" (for complete curl http calls logs)
`quiet`	Logical (`FALSE` by default). Do you want to suppress informative messages (not warnings)?
`...`	any other arguments for parallel downloading (more information at ZenodoRecord, `downloadFiles()` documentation)

Examples

```
## Not run:
 #simple download (sequential)
 download_zenodo("10.5281/zenodo.2547036")

 library(parallel)
```

```
#download files as parallel using a cluster approach (for both Unix/Win systems)
download_zenodo("10.5281/zenodo.2547036",
  parallel = TRUE, parallel_handler = parLapply, cl = makeCluster(2))

#download files as parallel using mclapply (for Unix systems)
download_zenodo("10.5281/zenodo.2547036",
  parallel = TRUE, parallel_handler = mclapply, mc.cores = 2)

## End(Not run)
```

`get_versions` *get_versions*

Description

`get_versions` allows to execute a workflow

Usage

```
get_versions(doi, logger = NULL)
```

Arguments

`doi`	a Zenodo DOI or concept DOI
`logger`	a logger to print messages. The logger can be either NULL, "INFO" (with minimum logs), or "DEBUG" (for complete curl http calls logs)

Value

an object of class `data.frame` giving the record versions including date, version number and version-specific DOI.

Examples

```
## Not run:
get_versions("10.5281/zenodo.2547036")

## End(Not run)
```

zen4R *Interface to 'Zenodo' REST API*

Description

Provides an Interface to 'Zenodo' (<https://zenodo.org>) REST API, including management of depositions, attribution of DOIs by 'Zenodo', upload and download of files.

Details

Package:	zen4R
Type:	Package
Version:	0.5
Date:	2021-09-14
License:	MIT
LazyLoad:	yes

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

zen4RLogger *zen4RLogger*

Description

zen4RLogger

Format

R6Class object.

Value

Object of R6Class for modelling a simple logger

Abstract Methods

`INFO(text)` Logger to report information. Used internally

`WARN(text)` Logger to report warnings. Used internally

`ERROR(text)` Logger to report errors. Used internally

Note

Logger class used internally by zen4R

`ZenodoManager` *ZenodoManager*

Description

ZenodoManager

Format

`R6Class` object.

Value

Object of `R6Class` for modelling an ZenodoManager

Methods

`new(url, token, logger, keyring_backend)` This method is used to instantiate the `ZenodoManager`. By default, the url is set to "https://zenodo.org/api". For tests, the Zenodo sandbox API URL can be used: https://sandbox.zenodo.org/api .

The token is mandatory in order to use Zenodo API deposit actions. By default, **zen4R** will first try to get it from environment variable 'ZENODO_PAT'.

The `keyring_backend` can be set to use a different backend for storing the Zenodo token with **keyring** (Default value is 'env').

The logger can be either NULL, "INFO" (with minimum logs), or "DEBUG" (for complete curl http calls logs)

`getToken()` Get Zenodo user token.

`getLicenses(pretty)` Get the list of licenses. By default the argument `pretty` is set to `TRUE` which will returns the list of licenses as `data.frame`. Set `pretty = FALSE` to get the raw list of licenses.

`getLicenseById(id)` Get license by Id

`getCommunities(pretty)` Get the list of communities. By default the argument `pretty` is set to `TRUE` which will returns the list of communities as `data.frame`. Set `pretty = FALSE` to get the raw list of communities.

`getCommunityById(id)` Get community by Id

`getGrants(pretty)` Get the list of grants. By default the argument `pretty` is set to `TRUE` which will returns the list of grants as `data.frame`. Set `pretty = FALSE` to get the raw list of grants.

`getGrantById(id)` Get grant by Id

`getFunders(pretty)` Get the list of funders. By default the argument `pretty` is set to `TRUE` which will returns the list of funders as `data.frame`. Set `pretty = FALSE` to get the raw list of funders.

`getFunderById(id)` Get funder by Id

`getDepositions(q, size, all_versions, exact, quiet)` Get the list of Zenodo records deposited in your Zenodo workspace. By defaut the list of depositions will be returned by page with a size of 10 results per page (default size of the Zenodo API). The parameter `q` allows to specify an ElasticSearch-compliant query to filter depositions (default query is empty to retrieve all records). The argument `all_versions`, if set to TRUE allows to get all versions of records as part of the depositions list. The argument `exact` specifies that an exact matching is wished, in which case paginated search will be disabled (only the first search page will be returned). Examples of ElasticSearch queries for Zenodo can be found at https://help.zenodo.org/guides/search/.

`getDepositionByConceptDOI(conceptdoi)` Get a Zenodo deposition record by concept DOI (generic DOI common to all deposition record versions)

`getDepositionByDOI(doi)` Get a Zenodo deposition record by DOI.

`getDepositionById(recid)` Get a Zenodo deposition record by its Zenodo specific record id.

`getDepositionByConceptId(conceptrecid)` Get a Zenodo deposition record by its Zenodo concept id.

`depositRecord(record, publish)` A method to deposit/update a Zenodo record. The record should be an object of class `ZenodoRecord`. The method returns the deposited record of class `ZenodoRecord`. The parameter `publish` (default value is `FALSE`) can be set to `TRUE` (to use CAUTIOUSLY, only if you want to publish your record)

`depositRecordVersion(record, publish)` A method to deposit a new version for a published record. For details about the behavior of this function, see https://developers.zenodo.org/#new-version

`deleteRecord(recordId)` Deletes a Zenodo record based on its identifier.

`deleteRecordByDOI(doi)` Deletes a Zenodo record based on its DOI. This DOI is necessarily a pre-reserved DOI corresponding to a draft record, and not a published DOI, as Zenodo does not allow to delete a record already published.

`deleteRecords(q)` Deletes all Zenodo deposited (unpublished) records. The parameter `q` allows to specify an ElasticSearch-compliant query to filter depositions (default query is empty to retrieve all records). Examples of ElasticSearch queries for Zenodo can be found at https://help.zenodo.org/guides/search/

`createEmptyRecord()` Creates an empty record in the Zenodo deposit. Returns the record newly created in Zenodo, as an object of class `ZenodoRecord` with an assigned identifier.

`editRecord(recordId)` Unlocks a record already submitted. This is required to edit metadata of a Zenodo record already published.

`discardChanges(recordId)` Discards changes operated on a record.

`publishRecord(recordId)` Publishes a deposited record online.

`getFiles(recordId)` Get the list of uploaded files for a deposited record

`uploadFile(path, record, recordId)` Uploads a file for a given Zenodo deposited record

`deleteFile(recordId, fileId)` Deletes a file for a given Zenodo deposited record

`getRecords(q, size, all_versions, exact)` Get the list of Zenodo records. By defaut the list of records will be returned by page with a size of 10 results per page (default size of the Zenodo API). The parameter `q` allows to specify an ElasticSearch-compliant query to filter depositions (default query is empty to retrieve all records). The argument `all_versions`, if

set to TRUE allows to get all versions of records as part of the depositions list. The argument `exact` specifies that an exact matching is wished, in which case paginated search will be disabled (only the first search page will be returned). Examples of ElasticSearch queries for Zenodo can be found at https://help.zenodo.org/guides/search/.

`getRecordByConceptDOI(conceptdoi)` Get a Zenodo published record by concept DOI (generic DOI common to all record versions)

`getRecordByDOI(doi)` Get a Zenodo published record by DOI.

`getRecordById(recid)` Get a Zenodo published record by its Zenodo specific record id.

`getRecordByConceptId(conceptrecid)` Get a Zenodo published record by its Zenodo concept id.

Note

Main user class to be used with **zen4R**

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
  ZENODO <- ZenodoManager$new(
     url = "https://sandbox.zenodo.org/api",
     token = "<your_token>",
     logger = "INFO"
  )

  #create (deposit) an empty record
  newRec <- ZENODO$createEmptyRecord()

  #create and fill a local (not yet deposited) record
  myrec <- ZenodoRecord$new()
  myrec$setTitle("my R package")
  myrec$setDescription("A description of my R package")
  myrec$setUploadType("software")
  myrec$addCreator(
    firstname = "John", lastname = "Doe",
    affiliation = "Independent", orcid = "0000-0000-0000-0000"
   )
  myrec$setLicense("mit")
  myrec$setAccessRight("open")
 myrec$setDOI("mydoi") #use this method if your DOI has been assigned elsewhere, outside Zenodo
  myrec$addCommunity("ecfunded")

  #deposit the record
  myrec <- ZENODO$depositRecord(myrec)

  #publish a record (with caution!!)
  #this method will PUBLISH the deposition done earlier
```

```
ZENODO$publishRecord(myrec$id)
#With even more caution the publication can be done with a shortcut argument at deposit time
ZENODO$depositRecord(myrec, publish = TRUE)

#delete a record (by id)
#this methods only works for unpublished deposits
#(if a record is published, it cannot be deleted anymore!)
ZENODO$deleteRecord(myrec$id)

#HOW TO UPLOAD FILES to a deposit

#upload a file
ZENODO$uploadFile("path/to/your/file", record = myrec)

#list files
zen_files <- ZENODO$getFiles(myrec$id)

#delete a file?
ZENODO$deleteFile(myrec$id, zen_files[[1]]$id)

## End(Not run)
```

ZenodoRecord *ZenodoRecord*

Description

ZenodoRecord

Format

R6Class object.

Value

Object of R6Class for modelling an ZenodoRecord

Methods

`new()` This method is used to instantiate a Zenodo Record

`prereserveDOI(prereserve)` Set prereserve_doi if `TRUE`, `FALSE` otherwise to create a record without prereserved DOI by Zenodo. By default, this method will be called to prereserve a DOI assuming the record created doesn't yet handle a DOI. To avoid prereserving a DOI call `$prereserveDOI(FALSE)` on your record.

`setUploadType(uploadType)` Set the upload type (mandatory). Value should be among the following: 'publication', 'poster','presentation','dataset','image','video', or 'software'

`setPublicationType(publicationType)` Set the publication type (mandatory if upload type is 'publication'). Value should be among the following: 'book','section','conferencepaper','article','patent','preprint', 'report','softwaredocumentation','thesis','technicalnote','workingpaper', or 'other'

`setImageType(imageType)` Set the image type (mandatory if image type is 'image'). Value should be among the following: 'figure','plot','drawing','diagram','photo', or 'other'

`setPublicationDate(publicationDate)` Set the publication date, as object of class `Date`

`setEmbargoDate(embargoDate)` Set the embargo date, as object of class `Date`

`setTitle(title)` Set title

`setDescription(description)` Set description

`setAccessRight(accessRight)` Set the access right. Value should be among the following: 'open','embargoed', 'restricted','closed'

`setAccessConditions(accessConditions)` Set access conditions.

`addCreator(firsname, lastname, name, affiliation, orcid, gnd)` Add a creator for the record. One approach is to use the `firstname` and `lastname` arguments, that by default will be concatenated for Zenodo as `lastname,firstname`. For more flexibility over this, the `name` argument can be directly used.

`removeCreator(by,property)` Removes a creator by a property. The `by` parameter should be the name of the creator property ('name' - in the form 'lastname, firstname', 'affiliation', 'orcid' or 'gnd'). Returns `TRUE` if some creator was removed, `FALSE` otherwise.

`removeCreatorByName(name)` Removes a creator by name. Returns `TRUE` if some creator was removed, `FALSE` otherwise.

`removeCreatorByAffiliation(affiliation)` Removes a creator by affiliation. Returns `TRUE` if some creator was removed, `FALSE` otherwise.

`removeCreatorByORCID(orcid)` Removes a creator by ORCID. Returns `TRUE` if some creator was removed, `FALSE` otherwise.

`removeCreatorByGND(gnd)` Removes a creator by GND. Returns `TRUE` if some creator was removed, `FALSE` otherwise.

`addContributor(firsname, lastname, type, affiliation, orcid, gnd)` Add a contributor for the record. Firstname, lastname, and type are mandatory. The type should be an object of class `character` among values: ContactPerson, DataCollector, DataCurator, DataManager, Distributor, Editor, Funder, HostingInstitution, Producer, ProjectLeader, ProjectManager, ProjectMember, RegistrationAgency, RegistrationAuthority, RelatedPerson, Researcher, ResearchGroup, RightsHolder, Supervisor, Sponsor, WorkPackageLeader, Other.

`removeContributor(by,property)` Removes a contributor by a property. The `by` parameter should be the name of the contributor property ('name' - in the form 'lastname, firstname', 'affiliation', 'orcid' or 'gnd'). Returns `TRUE` if some contributor was removed, `FALSE` otherwise.

`removeContributorByName(name)` Removes a contributor by name. Returns `TRUE` if some contributor was removed, `FALSE` otherwise.

`removeContributorByAffiliation(affiliation)` Removes a contributor by affiliation. Returns `TRUE` if some contributor was removed, `FALSE` otherwise.

`removeContributorByORCID(orcid)` Removes a contributor by ORCID. Returns `TRUE` if some contributor was removed, `FALSE` otherwise.

`removeContributorByGND(gnd)` Removes a contributor by GND. Returns `TRUE` if some contributor was removed, `FALSE` otherwise.

`setLicense(licenseId)` Set license. The license should be set with the Zenodo id of the license. If not recognized by Zenodo, the function will return an error. The list of licenses can fetched with the `ZenodoManager` and the function `$getLicenses()`.

`setDOI(doi)` Set the DOI. This method can be used if a DOI has been already assigned outside Zenodo. This method will call the method `$prereserveDOI(FALSE)`.

`getConceptDOI()` Get the concept (generic) DOI. The concept DOI is a generic DOI common to all versions of a Zenodo record. When a deposit is unsubmitted, this concept DOI is inherited based on the prereserved DOI of the first record version.

`getFirstDOI()` Get DOI of the first record version.

`getLastDOI()` Get DOI of the latest record version.

`getVersions()` Get a `data.frame` listing record versions with creation/publication date, version (ordering number) and DOI.

`setVersion(version)` Set the version.

`setLanguage(language)` Set the language ISO 639-2 or 639-3 code.

`addRelatedIdentifier(relation, identifier)` Adds a related identifier with a given relation. Relation can be one of among following values: isCitedBy, cites, isSupplementTo, isSupplementedBy, isNewVersionOf, isPreviousVersionOf, isPartOf, hasPart, compiles, isCompiledBy, isIdenticalTo, isAlternateIdentifier

coderemoveRelatedIdentifier(relation, identifier) Remove a related identifier

`setReferences(references)` Set a vector of character strings as references

`addReference(reference)` Adds a reference to the record metadata. Return `TRUE` if added, `FALSE` otherwise.

`removedReference(reference)` Removes a reference from the record metadata. Return `TRUE` if removed, `FALSE` otherwise.

`setKeywords(keywords)` Set a vector of character strings as keywords

`addKeyword(keyword)` Adds a keyword to the record metadata. Return `TRUE` if added, `FALSE` otherwise.

`removedKeyword(keyword)` Removes a keyword from the record metadata. Return `TRUE` if removed, `FALSE` otherwise.

`addSubject(term, identifier)` Add a Subject for the record.

`removeSubject(by,property)` Removes a subject by a property. The by parameter should be the name of the subject property ('term' or 'identifier'). Returns `TRUE` if some subject was removed, `FALSE` otherwise.

`removeSubjectByTerm(term)` Removes a subject by term. Returns `TRUE` if some subject was removed, `FALSE` otherwise.

`removeSubjectByIdentifier(identifier)` Removes a subject by identifier. Returns `TRUE` if some subject was removed, `FALSE` otherwise.

`setNotes(notes)` Set notes. HTML is not allowed

`setCommunities(communities)` Set a vector of character strings identifying communities

`addCommunity(community)` Adds a community to the record metadata. Return `TRUE` if added, `FALSE` otherwise. The community should be set with the Zenodo id of the community. If not recognized by Zenodo, the function will return an error. The list of communities can fetched with the `ZenodoManager` and the function `$getCommunities()`.

`removedCommunity(community)` Removes a community from the record metadata. Return `TRUE` if removed, `FALSE` otherwise.

`setGrants(grants)` Set a vector of character strings identifying grants

`addGrant(grant)` Adds a grant to the record metadata. Return `TRUE` if added, `FALSE` otherwise. The grant should be set with the id of the grant. If not recognized by Zenodo, the function will return an warning only. The list of grants can fetched with the `ZenodoManager` and the function `$getGrants()`.

`removeGrant(grant)` Removes a grant from the record metadata. Return `TRUE` if removed, `FALSE` otherwise.

`setJournalTitle(title)` Set Journal title, object of class `character`, if deposition is a published article.

`setJournalVolume(volume)` Set Journal volume, object of class `character`, if deposition is a published article.

`setJournalIssue(issue)` Set Journal issue, object of class `character`, if deposition is a published article.

`setJournalPages(pages)` Set Journal pages, object of class `character`, if deposition is a published article.

`setConferenceTitle(title)` Set Conference title, object of class `character`.

`setConferenceAcronym(acronym)` Set conference acronym, object of class `character`.

`setConferenceDates(dates)` Set conference dates, object of class `character`.

`setConferencePlace(place)` Set conference place, object of class `character`, in the format city, country (e.g. Kingston, Jamaica). Conference title or acronym must also be specified if this field is specified.

`setConferenceUrl(url)` Set conference url, object of class `character`.

`setConferenceSession(session)` Set conference session, object of class `character`.

`setConferenceSessionPart(part)` Set conference session part, object of class `character`.

`setImprintPublisher(publisher)` Set imprint publisher, object of class `character`.

`setImprintISBN(isbn)` Set imprint ISBN, object of class `character`.

`setImprintPlace(place)` Set imprint place, object of class `character`.

`setPartofTitle(title)` Set the book title in case of a chapter, object of class `character`.

`setPartofPages(pages)` Set the page numbers of book, object of class `character`.

`setThesisUniversity(university)` Set the thesis university, object of class `character`

`addThesisSupervisor(firsname, lastname, affiliation, orcid, gnd)` Add a thesis supervisor for the record.

`removeThesisSupervisor(by,property)` Removes a thesi supervisor by a property. The by parameter should be the name of the thesis supervisor property ('name' - in the form 'lastname, firstname', 'affiliation', 'orcid' or 'gnd'). Returns `TRUE` if some thesis supervisor was removed, `FALSE` otherwise.

`removeThesisSupervisorByName(name)` Removes a thesis supervisor by name. Returns `TRUE` if some thesis supervisor was removed, `FALSE` otherwise.

`removeThesisSupervisorByAffiliation(affiliation)` Removes a thesis supervisor by affiliation. Returns `TRUE` if some thesis supervisor was removed, `FALSE` otherwise.

`removeThesisSupervisorByORCID(orcid)` Removes a thesis supervisor by ORCID. Returns `TRUE` if some thesis supervisor was removed, `FALSE` otherwise.

`removeThesisSupervisorByGND(gnd)` Removes a thesis supervisor by GND. Returns `TRUE` if some thesis supervisor was removed, `FALSE` otherwise.

`exportAs(format, filename)` Export metadata in a format supported by Zenodo. Supported formats are: BibTeX, CSL, DataCite, DublinCore, DCAT, JSON, JSON-LD, GeoJSON, MARCXML. The exported will be named with the specified filename followed by the format name.

`exportAsBibTeX(filename)` Export metadata as BibTeX (BIB) file

`exportAsCSL(filename)` Export metadata as CSL (JSON) file

`exportAsDataCite(filename)` Export metadata as DataCite (XML) file

`exportAsDublinCore(filename)` Export metadata as Dublin Core file

`exportAsDCAT(filename)` Export metadata as DCAT (RDF) file

`exportAsJSON(filename)` Export metadata as JSON file

`exportAsJSONLD(filename)` Export metadata as JSON-LD (JSON) file

`exportAsGeoJSON(filename)` Export metadata as GeoJSON (JSON) file

`exportAsMARCXML{filename}` Export metadata as MARCXML (XML) file

`exportAsAllFormats(filename)` Export metadata as all Zenodo supported metadata formats. THis function will create one file per Zenodo metadata formats.

`listFiles(pretty)` List files attached to the record. By default `pretty` is TRUE and the output will be a `data.frame`, otherwise a `list` will be returned.

`downloadFiles(path, files, parallel, parallel_handler, cl, ...)` Download files attached to the record. The `path` can be specified as target download directory (by default it will be the current working directory).

Download can be restrained to one more file which names can be provided as vector using the `files` argument. By default, all files are downloaded.

The argument `parallel` (default is `FALSE`) can be used to parallelize the files download. If set to `TRUE`, files will be downloaded in parallel. using the chosen `parallel_handler`, eg `mclapply` interface from **parallel** package. To use a different parallel handler (such as eg `parLapply` or `parSapply`), specify its function in `parallel_handler` argument. For cluster-based parallel download, this is the way to proceed. In that case, the cluster should be created earlier by the user with `makeCluster` and passed as `cl` argument. After downloading all files, the cluster will be stopped automatically. See examples in `download_zenodo` utility function.

The logical argument `quiet` (default is `FALSE`) can be set to suppress informative messages (not warnings).

Additional arguments inherited from `parallel::mclapply` or the custom `parallel_handler` can be added (eg. `mc.cores` for `mclapply`)

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

ZenodoRequest *ZenodoRequest*

Description

ZenodoRequest

Format

R6Class object.

Value

Object of R6Class for modelling a generic Zenodo request

Methods

`new(url, type, request, data, file, token, logger)` This method is used to instantiate a object for doing an Zenodo request

`getRequest()` Get the request payload

`getRequestHeaders()` Get the request headers

`getStatus()` Get the request status code

`getResponse()` Get the request response

`getException()` Get the exception (in case of request failure)

`getResult()` Get the result `TRUE` if the request is successful, `FALSE` otherwise

Note

Abstract class used internally by **zen4R**

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Index