Package 'rdatacite'

March 4, 2020

Type Package

Title Client for the 'DataCite' API

Description Client for the web service methods provided by 'DataCite' (<https://www.datacite.org/>), including functions to interface with their 'RESTful' search API. The API is backed by 'Elasticsearch', allowing expressive queries, including faceting.

Version 0.5.2

License MIT + file LICENSE

URL https://docs.ropensci.org/rdatacite, https://github.com/ropensci/rdatacite

BugReports https://github.com/ropensci/rdatacite/issues

Encoding UTF-8

Imports jsonlite, crul (>= 0.7.4), tibble

Suggests knitr, testthat, webmockr, vcr

RoxygenNote 7.0.2

X-schema.org-applicationCategory Data

X-schema.org-keywords data, scholarly, dataset, https, API, web-services

X-schema.org-isPartOf https://ropensci.org

NeedsCompilation no

Author Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>)

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2020-03-04 05:50:06 UTC

R topics documented:

`rdatacite-package` *DataCite R client*

Description

DataCite R client

HTTP Requests

All HTTP requests are GET requests, and are sent with the following headers:

- Accept: application/vnd.api+json; version=2
- User-Agent: r-curl/4.3 crul/0.9.0 rOpenSci(rdatacite/0.5.0)
- X-USER-AGENT: r-curl/4.3 crul/0.9.0 rOpenSci(rdatacite/0.5.0)

The user-agent strings change as the versions of each package change.

Methods in the package

- `dc_providers()`
- `dc_reports()`
- `dc_check()`
- `dc_events()`
- `dc_dois()`
- `dc_clients()`
- `dc_client_prefixes()`
- `dc_provider_prefixes()`
- `dc_status()`
- `dc_prefixes()`
- `dc_activities()`

rdatacite defunct functions

- dc_data_center
- dc_data_centers
- dc_facet
- dc_member
- dc_members
- dc_mlt
- dc_oai_getrecord
- dc_oai_identify
- dc_oai_listidentifiers
- dc_oai_listmetadataformats
- dc_oai_listrecords
- dc_oai_listsets
- dc_search
- dc_stats
- dc_work
- dc_works

Content negotation

For content negotation see `rcrossref::cr_cn()`, which can be used for Crossref, DataCite and Medra DOIs

GraphGL API

rdatacite does not support the GraphGL API https://support.datacite.org/docs/datacite-graphql-api-guide - we suggest trying the `ghql` package (https://github.com/ropensci/ghql/)

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

---

dc_activities *DataCite REST API: activities*

Description

DataCite REST API: activities

Usage

```
dc_activities(
  ids = NULL,
  query = NULL,
  limit = 25,
  page = 1,
  cursor = NULL,
  ...
)
```

Arguments

ids	(character) one or more activity IDs
query	(character) Query string
limit	(numeric/integer) results per page
page	(numeric/integer) the page to get results for. default: 1
cursor	(character) page cursor (used instead of `limit` param). to use cursor pagination, set `cursor = 1`, then use the link in $links$next
...	curl options passed on to crul::verb-GET

Details

for more info on the /activities route see https://support.datacite.org/docs/tracking-provenance

Examples

```
## Not run:
if (dc_check()) {
x <- dc_activities()
x
# dc_activities(x$data$id[1]) # FIXME: doesn't work, returns no data
# dc_activities(query = "ecology") # FIXME: this thlimit a 500 error
}
## End(Not run)
```

---

dc_clients *DataCite REST API: clients*

Description

DataCite REST API: clients

Usage

```
dc_clients(
  ids = NULL,
  query = NULL,
  year = NULL,
  provider_id = NULL,
  software = NULL,
  include = NULL,
  limit = 25,
  page = 1,
  cursor = NULL,
  ...
)
```

Arguments

`ids`	(character) one or more client IDs
`query`	(character) Query string
`year`	(integer/numeric/character) a year
`provider_id`	a provider ID
`software`	no idea what should go here, anyone?
`include`	(character) vector of fields to return
`limit`	(numeric/integer) results per page
`page`	(numeric/integer) the page to get results for. default: 1
`cursor`	(character) page cursor (used instead of `limit` param). to use cursor pagination, set `cursor = 1`, then use the link in $links$next
`...`	curl options passed on to crul::verb-GET

Examples

```
## Not run:
if (dc_check()) {
x <- dc_clients()
x
dc_clients(x$data$id[1])
dc_clients(x$data$id[1:2], verbose = TRUE)
}
## End(Not run)
```

`dc_client_prefixes` *DataCite REST API: client prefixes*

Description

DataCite REST API: client prefixes

Usage

```
dc_client_prefixes(
  query = NULL,
  year = NULL,
  client_id = NULL,
  prefix_id = NULL,
  sort = NULL,
  include = NULL,
  limit = 25,
  page = 1,
  cursor = NULL,
  ...
)
```

Arguments

`query`	(character) Query string
`year`	(integer/numeric/character) a year
`client_id`	a client ID
`prefix_id`	a prefix ID
`sort`	(character) variable to sort by
`include`	(character) vector of fields to return
`limit`	(numeric/integer) results per page
`page`	(numeric/integer) the page to get results for. default: 1
`cursor`	(character) page cursor (used instead of `limit` param). to use cursor pagination, set `cursor = 1`, then use the link in $links$next
`...`	curl options passed on to crul::verb-GET

Examples

```
## Not run:
if (dc_check()) {
x <- dc_client_prefixes()
x
}
## End(Not run)
```

dc_dois *DataCite REST API: dois*

Description

DataCite REST API: dois

Usage

```
dc_dois(
  ids = NULL,
  query = NULL,
  created = NULL,
  registered = NULL,
  provider_id = NULL,
  client_id = NULL,
  person_id = NULL,
  resource_type_id = NULL,
  subject = NULL,
  schema_version = NULL,
  random = NULL,
  sample_size = NULL,
  sample_group = NULL,
  include = NULL,
  sort = NULL,
  limit = 25,
  page = 1,
  cursor = NULL,
  ...
)
```

Arguments

ids	(character) one or more DOIs
query	(character) Query string. See Querying below.
created	(character) metadata where year of DOI creation is created. See Filtering Responses below.
registered	(character) metadata where year of DOI registration is year. See Filtering Responses below.
provider_id	(character) metadata associated with a specific DataCite provider. See Filtering Responses below.
client_id	(character) metadata associated with a specific DataCite client. See Filtering Responses below.
person_id	(character) metadata associated with a specific person's ORCID iD. See Filtering Responses below.

resource_type_id	(character) metadata for a specific resourceTypeGeneral. See Filtering Responses below.
subject	(character)
schema_version	(character) metadata where schema version of the deposited metadata is schema-version. See Filtering Responses below.
random	(logical) return random set of results, can be combined with any kind of query. default: FALSE.
sample_size	(character)
sample_group	(character)
include	(character) vector of fields to return
sort	(character) variable to sort by
limit	(numeric/integer) results per page
page	(numeric/integer) the page to get results for. default: 1
cursor	(character) page cursor (used instead of limit param). to use cursor pagination, set cursor = 1, then use the link in $links$next
...	curl options passed on to crul::verb-GET

Querying

See https://support.datacite.org/docs/api-queries for details

Filtering Responses

See https://support.datacite.org/docs/api-queries#section-filtering-list-responses for details

Examples

```
## Not run:
if (dc_check()) {
x <- dc_dois()
x
dc_dois(query = "birds")
dc_dois(query = "climate change")
dc_dois(query = "publicationYear:2016")
x <- dc_dois(query = "creators.familyName:mil*", verbose = TRUE)
lapply(x$data$attributes$creators, "[[", "familyName")
x <- dc_dois(query = "titles.title:climate +change")
lapply(x$data$attributes$titles, "[[", "title")
dc_dois(client_id = "dryad.dryad")
dc_dois(x$data$id[1])
dc_dois(x$data$id[1:3])
dc_dois("10.5281/zenodo.1308060")

# pagination
dc_dois(limit = 1)
x <- dc_dois(cursor = 1)
x$links$`next`
```

```
}
## End(Not run)
```

---

dc_events *DataCite REST API: events*

Description

DataCite REST API: events

Usage

```
dc_events(
  ids = NULL,
  query = NULL,
  subj_id = NULL,
  obj_id = NULL,
  doi = NULL,
  orcid = NULL,
  prefix = NULL,
  subtype = NULL,
  subject = NULL,
  source_id = NULL,
  registrant_id = NULL,
  relation_type_id = NULL,
  issn = NULL,
  publication_year = NULL,
  year_month = NULL,
  include = NULL,
  sort = NULL,
  limit = 25,
  page = 1,
  cursor = NULL,
  ...
)
```

Arguments

ids	(character) one or more event IDs
query	(character) Query for any event information
subj_id	(character) The identifier for the event subject, expressed as a URL. For example: https://doi.org/10.7272/q6qn64nk
obj_id	(character) The identifier for the event object, expressed as a URL. For example: https://doi.org/10.7272/q6qn64nk
doi	(character) The subj-id or obj-id of the event, expressed as a DOI. For example: 10.7272/q6qn64nk

orcid	(character) an ORCID, presumably
prefix	(character) The DOI prefix of the subj-id or obj-id of the event. For example: 10.7272
subtype	(character) xxx
subject	(character) xxx
source_id	(character) a source ID. See Details
registrant_id	(character)
relation_type_id	(character) a relation-type ID. See Details
issn	(character) an ISSN, presumably
publication_year	(character) the publication year
year_month	(character) The year and month in which the event occurred, in the format YYYY-MM. For example 2018-08
include	(character) vector of fields to return
sort	(character) variable to sort by
limit	(numeric/integer) results per page
page	(numeric/integer) the page to get results for. default: 1
cursor	(character) page cursor (used instead of limit param). to use cursor pagination, set cursor = 1, then use the link in $links$next
...	curl options passed on to crul::verb-GET

Details

See https://support.datacite.org/docs/eventdata-guide for details on possible values for parameters

Examples

```
## Not run:
if (dc_check()) {
# dc_events(query = "birds")
}
## End(Not run)
```

---

dc_prefixes *DataCite REST API: prefixes*

Description

DataCite REST API: prefixes

Usage

```
dc_prefixes(include = NULL, limit = 25, page = 1, cursor = NULL, ...)
```

Arguments

include	(character) vector of fields to return
limit	(numeric/integer) results per page
page	(numeric/integer) result page, the record to start at
cursor	(character) page cursor (used instead of limit param)
...	curl options passed on to crul::HttpClient

Examples

```
## Not run:
if (dc_check()) {
x <- dc_prefixes()
x
dc_prefixes(limit = 3)
}
## End(Not run)
```

---

dc_providers *DataCite REST API: providers*

Description

DataCite REST API: providers

Usage

```
dc_providers(
  ids = NULL,
  query = NULL,
  year = NULL,
  region = NULL,
  organization_type = NULL,
  focus_area = NULL,
  include = NULL,
  limit = 25,
  page = 1,
  cursor = NULL,
  ...
)
```

Arguments

ids	(character) one or more provider IDs
query	(character) query string
year	(character) year

region	(character) region name
organization_type	(character) organization type
focus_area	(character) focus area
include	(character) vector of fields to return
limit	(numeric/integer) results per page
page	(numeric/integer) result page, the record to start at
cursor	(character) page cursor (used instead of limit param)
...	curl options passed on to crul::HttpClient

Examples

```
## Not run:
if (dc_check()) {
x <- dc_providers()
x
dc_providers(limit = 3)
dc_providers(ids = x$data$id[1:5])
}
## End(Not run)
```

---

dc_provider_prefixes *DataCite REST API: provider prefixes*

Description

DataCite REST API: provider prefixes

Usage

```
dc_provider_prefixes(include = NULL, limit = 25, page = 1, cursor = NULL, ...)
```

Arguments

include	(character) vector of fields to return
limit	(numeric/integer) results per page
page	(numeric/integer) result page, the record to start at
cursor	(character) page cursor (used instead of limit param)
...	curl options passed on to crul::HttpClient

Examples

```
## Not run:
if (dc_check()) {
x <- dc_provider_prefixes()
x
dc_provider_prefixes(limit = 3)
}
## End(Not run)
```

`dc_reports` *DataCite REST API: reports*

Description

DataCite REST API: reports

Usage

```
dc_reports(
  ids = NULL,
  platform = NULL,
  report_name = NULL,
  report_id = NULL,
  release = NULL,
  created = NULL,
  created_by = NULL,
  include = NULL,
  limit = 25,
  page = 1,
  ...
)
```

Arguments

`ids`	(character) one or more report IDs
`platform`	(character) Name of the Platform the usage is being requested for. This can be omitted if the service provides usage for only one platform.
`report_name`	(character) The long name of the report
`report_id`	(character) The report ID or code or shortname. Typically this will be the same code provided in the Report parameter of the request
`release`	(character) The release or version of the report
`created`	(character) Time the report was prepared. Format as defined by date-time - RFC3339
`created_by`	(character) Name of the organization producing the report
`include`	(character) vector of fields to return

limit	(numeric/integer) results per page
page	(numeric/integer) result page, the record to start at
...	curl options passed on to crul::HttpClient

Examples

```
## Not run:
if (dc_check()) {
x <- dc_reports()
x
dc_reports(created = "2019-08-01T07:00:00.000Z")
dc_reports(created_by = "urn:node:GOA")
dc_reports(limit = 3)
# dc_reports(ids = x$reports$id[1:3]) # FIXME: doesn't work
}
## End(Not run)
```

---

dc_status *DataCite REST API: status of the API*

Description

DataCite REST API: status of the API

Usage

```
dc_status(...)
```

Arguments

...	curl options passed on to crul::HttpClient

Examples

```
## Not run:
if (dc_check()) {
dc_status()
}
## End(Not run)
```

Index