

# Package: rdatacite (via r-universe)

November 27, 2024

**Type** Package

**Title** Client for the 'DataCite' API

**Version** 0.5.4

**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.

**License** MIT + file LICENSE

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

**BugReports** <https://github.com/ropensci/rdatacite/issues>

**Imports** crul (>= 0.7.4), jsonlite, tibble, utils

**Suggests** testthat, vcr

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.2

**X-schema.org-applicationCategory** Data

**X-schema.org-isPartOf** <https://ropensci.org>

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

**Config/pak/sysreqs** libssl-dev

**Repository** <https://ropensci.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci/rdatacite>

**RemoteRef** master

**RemoteSha** b86e18346d60ebaa279c047e6d27276051aed282

## Contents

rdatacite-package . . . . .	2
dc_activities . . . . .	3
dc_clients . . . . .	4
dc_client_prefixes . . . . .	6
dc_cn . . . . .	7
dc_dois . . . . .	8
dc_events . . . . .	10
dc_prefixes . . . . .	11
dc_providers . . . . .	12
dc_provider_prefixes . . . . .	13
dc_reports . . . . .	14
dc_status . . . . .	15
<b>Index</b>	<b>16</b>

---

rdatacite-package	<i>rdatacite</i>
-------------------	------------------

---

## 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 negotiation**

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

**GraphQL API**

rdatacite does not support the GraphQL 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](mailto: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 <a href="#">crul::verb-GET</a>

**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 <code>\$links\$next</code>
...	curl options passed on to <a href="#">crul::verb-GET</a>

**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 <code>limit</code> param). to use cursor pagination, set <code>cursor = 1</code> , then use the link in <code>\$links\$next</code>
...	curl options passed on to <a href="#">curl::verb-GET</a>

### Examples

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

---

dc\_cn *DataCite content negotiation*

---

## Description

DataCite content negotiation

## Usage

```
dc_cn(dois, format = "bibtex", style = "apa", locale = "en-US", ...)
```

## Arguments

dois	(character) one or more DOIs
format	Name of the format. One of "rdf-xml", "turtle", "citeproc-json", "schemaorg", "codemeta", "text", "ris", "bibtex" (default), "datacite-xml", "datacite-json", "bibentry", or "jats".
style	a CSL style (for text format only). See 'rcrossref::get_styles()' for options. Default: 'apa'. If there's a style that DataCite doesn't support you'll get a (500) Internal Server Error
locale	Language locale. See '?Sys.getlocale'
...	curl options passed on to <a href="#">crul::verb-GET</a>

## References

<https://support.datacite.org/docs/datacite-content-resolver>

## See Also

see also `rcrossref:::cr_cn` for a more general purpose content negotiation interface

## Examples

```
## Not run:
dc_cn("10.5281/zenodo.50213")
dc_cn(c("10.5281/zenodo.50213", "10.5281/zenodo.57081"), "text")
dc_cn(c("a-bad-doi", "10.5281/zenodo.50213", "10.5281/zenodo.57081"), "text")

## 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 <code>curl::verb-GET</code>

## 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: <a href="https://doi.org/10.7272/q6qn64nk">https://doi.org/10.7272/q6qn64nk</a>
obj_id	(character) The identifier for the event object, expressed as a URL. For example: <a href="https://doi.org/10.7272/q6qn64nk">https://doi.org/10.7272/q6qn64nk</a>
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 <a href="#">crul::verb-GET</a>

### 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 <a href="#">crul::HttpClient</a>

### 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 <a href="#">crul::HttpClient</a>

**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

## \* package

rdatacite-package, 2

crul::HttpClient, 12, 13, 15

crul::verb-GET, 4–7, 9, 11

dc\_activities, 3

dc\_activities(), 2

dc\_check(), 2

dc\_client\_prefixes, 6

dc\_client\_prefixes(), 2

dc\_clients, 4

dc\_clients(), 2

dc\_cn, 7

dc\_dois, 8

dc\_dois(), 2

dc\_events, 10

dc\_events(), 2

dc\_prefixes, 11

dc\_prefixes(), 2

dc\_provider\_prefixes, 13

dc\_provider\_prefixes(), 2

dc\_providers, 12

dc\_providers(), 2

dc\_reports, 14

dc\_reports(), 2

dc\_status, 15

dc\_status(), 2

rdatacite (rdatacite-package), 2

rdatacite-package, 2