Package: rcrossref (via r-universe)

November 27, 2024

Title Client for Various 'CrossRef' 'APIs'

Description Client for various 'CrossRef' 'APIs', including 'metadata' search with their old and newer search 'APIs', get 'citations' in various formats (including 'bibtex', 'citeproc-json', 'rdf-xml', etc.), convert 'DOIs' to 'PMIDs', and 'vice versa', get citations for 'DOIs', and get links to full text of articles when available.

Version 1.2.009

License MIT + file LICENSE

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

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

Roxygen list(markdown = TRUE)

Encoding UTF-8

Language en-US

Imports methods, utils, jsonlite (>= 1.5), crul (>= 0.7.4), xml2 (>= 1.1.1), plyr, dplyr (>= 0.7.4), tibble, R6, shiny, miniUI, stringr, DT, stats, rlang

Suggests roxygen2 (>= 7.1.0), testthat, knitr, rmarkdown, vcr (>= 0.2.6), bibtex, withr

RoxygenNote 7.2.3

X-schema.org-applicationCategory Literature

X-schema.org-keywords text-ming, literature, pdf, xml, publications, citations, full-text, TDM, Crossref

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

Config/pak/sysreqs make libicu-dev libxml2-dev libssl-dev zlib1g-dev

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

RemoteUrl https://github.com/ropensci/rcrossref

RemoteRef main

RemoteSha bf3fcfe0d7deede6c93847081f84e8ab45ba079e

Contents

`rcrossref-package` *rcrossref*

Description

R Client for Various CrossRef APIs.

Crossref APIs

rcrossref interacts with the main Crossref metadata search API at https://github.com/CrossRef/rest-api-doc, the old metadata search API at http://search.labs.crossref.org/, their DOI Content Negotiation service at http://citation.crosscite.org/docs.html, and the *Text and Data Mining* project http://tdmsupport.crossref.org/

Crossref's API issue tracker lives at https://gitlab.com/crossref/issues it's a good place to go ask them about things related to their API that go beyond the R interface here.

Defunct

See rcrossref-deprecated and rcrossref-defunct for details.

What am I actually searching?

When you use the `cr_*()` functions in this package, you are using the Crossref search API described at https://github.com/CrossRef/rest-api-doc When you search with query terms, on Crossref servers they are not searching full text, or even abstracts of articles, but only what is available in the data that is returned to you. That is, they search article titles, authors, etc. For some discussion on this, see https://gitlab.com/crossref/issues/issues/101

Rate limiting

From time to time Crossref needs to impose rate limits to ensure that the free API is usable by all. Any rate limits that are in effect will be advertised in the `X-Rate-Limit-Limit` and `X-Rate-Limit-Interval` HTTP headers.

This boils down to: they allow X number of requests per some time period. The numbers can change so we can't give a rate limit that will always be in effect. If you're curious pass in `verbose = TRUE` to your function call, and you'll get headers that will display these rate limits.

Be nice and share your email with Crossref

The Crossref team encourage requests with appropriate contact information and will forward you to a dedicated API cluster for improved performance when you share your email address with them. https://github.com/CrossRef/rest-api-doc#good-manners–more-reliable-service

To pass your email address to Crossref via this client, simply store it as environment variable in `.Renviron` like this:

1. Open file: `file.edit("~/.Renviron")`
2. Add email address to be shared with Crossref `crossref_email = name@example.com`
3. Save the file and restart your R session

Don't wanna share your email any longer? Simply delete it from `~/.Renviron`

Text mining

All Crossref specific text mining functions are now deprecated, and moved to a new package `crminer`.

Another package **fulltext** is designed solely to do general purpose text mining involving Crossref and other sources of scholarly metadata and full text.

High and Low Level APIs

For the Crossref search API (the functions `cr_funders()`, `cr_journals()`, `cr_licenses()`, `cr_members()`, `cr_prefixes()`, `cr_types()`, `cr_works()`), there is a high level API and a low level. The high level is accessible through those functions just listed (e.g., `cr_works()`), whereas the low level is accessible via the same fxn name with an underscore (e.g., `cr_works_()`). The high level API does data requests, and parses to data.frame's. Since the high level API functions have been around a while, we didn't want to break their behavior, so the low level API functions are separate, and only do the data request, giving back json or a list, with no attempt to parse any further. The low level API functions will be faster because there's much less parsing, and therefore less prone to potential errors due to changes in the Crossref API that could cause parsing errors. Note that cursor feature works with both high and low level.

RStudio Addin

On installation of **rcrossref** you get an RStudio Addin. To use the Addin, go to the top toolbar > Tools > Addins > Add Crossref Citations. You'll get a window pop up that you can put in DOIs for. If the DOI is found, the bibtex citations will be added to a file called `crossref.bib`. New citations will be appended to that file. Addin authored by Hao Zhu https://github.com/haozhu233

`cr_abstract` *Get abstract*

Description

Get abstract

Usage

```
cr_abstract(doi, ...)
```

Arguments

`doi`	(character) a DOI, required.
`...`	Named parameters passed on to `HttpClient`

Examples

```
## Not run:
# abstract found
cr_abstract('10.1109/TASC.2010.2088091')
cr_abstract("10.1175//2572.1")
cr_abstract("10.1182/blood.v16.1.1039.1039")

# doi not found
# cr_abstract(doi = '10.5284/1011335')

# abstract not found, throws error
# cr_abstract(doi = '10.1126/science.169.3946.635')

# a random DOI
# cr_abstract(cr_r(1))

## End(Not run)
```

cr_agency *Check the DOI minting agency on one or more dois*

Description

Check the DOI minting agency on one or more dois

Usage

```
cr_agency(dois = NULL, .progress = "none", ...)
```

Arguments

dois	(character) One or more article or organization dois.
.progress	Show a plyr-style progress bar? Options are "none", "text", "tk", "win", and "time". See create_progress_bar for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the cursor param. When using the cursor param, this argument only accept a boolean, either TRUE or FALSE; any non-boolean is coerced to FALSE.
...	Named parameters passed on to verb-GET

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

https://github.com/CrossRef/rest-api-doc

Examples

```
## Not run:
cr_agency(dois = '10.13039/100000001')
cr_agency(
  dois = c('10.13039/100000001','10.13039/100000015','10.5284/1011335'))

## End(Not run)
```

`cr_citation_count` *Get a citation count via CrossRef OpenURL*

Description

Get a citation count via CrossRef OpenURL

Usage

```
cr_citation_count(
  doi,
  url = "http://www.crossref.org/openurl/",
  key = "cboettig@ropensci.org",
  async = FALSE,
  ...
)
```

Arguments

`doi`	(character) One or more digital object identifiers. If `async=FALSE` we do synchronous HTTP requests in an `lapply` call, but if `async=TRUE`, we do asynchronous HTTP requests.
`url`	(character) the url for the function (should be left to default)
`key`	your Crossref OpenURL email address, either enter, or loads from `.Rprofile`. We use a default, so you don't need to pass this.
`async`	(logical) use async HTTP requests. Default: `FALSE`
`...`	Curl options passed on to `crul::HttpClient()`

Details

See https://www.crossref.org/labs/openurl/ for more info on this Crossref API service.

This number is also known as **cited-by**

Note that this number may be out of sync/may not match that that the publisher is showing (if they show it) for the same DOI/article.

We've contacted Crossref about this, and they have confirmed this. Unfortunately, we can not do anything about this.

I would imagine it's best to use this data instead of from the publishers, and this data you can get programatically :)

Value

a data.frame, with columns `doi` and `count`. The count column has numeric values that are the citation count for that DOI, or `NA` if not found or no count available

failure behavior

When a DOI does not exist, we may not get a proper HTTP status code to throw a proper stop status, so we grep on the text itself, and throw a stop if only one DOI passed and not using async, or warning if more than one DOI passed or if using async.

Author(s)

Carl Boettiger <cboettig@gmail.com>, Scott Chamberlain

See Also

cr_search(), cr_r()

Examples

```
## Not run:
cr_citation_count(doi="10.1371/journal.pone.0042793")
cr_citation_count(doi="10.1016/j.fbr.2012.01.001")
## many
dois <- c("10.1016/j.fbr.2012.01.001", "10.1371/journal.pone.0042793")
cr_citation_count(doi = dois)
# DOI not found
cr_citation_count(doi="10.1016/j.fbr.2012")

# asyc
dois <- c("10.1016/j.fbr.2012.01.001", "10.1371/journal.pone.0042793",
 "10.1016/j.fbr.2012", "10.1109/tsp.2006.874779", "10.1007/bf02231542",
 "10.1007/s00277-016-2782-z", "10.1002/9781118339893.wbeccp020",
 "10.1177/011542659200700105", "10.1002/chin.197444438",
 "10.1002/9781118619599.ch4", "10.1007/s00466-012-0724-8",
 "10.1017/s0376892900029477", "10.1167/16.12.824")
res <- cr_citation_count(doi = dois, async = TRUE)
## verbose curl
res <- cr_citation_count(doi = dois, async = TRUE, verbose = TRUE)
res
## time comparison
system.time(cr_citation_count(doi = dois, async = TRUE))
system.time(cr_citation_count(doi = dois, async = FALSE))

# from a set of random DOIs
cr_citation_count(cr_r(50), async = TRUE)

## End(Not run)
```

---

cr_cn *Get citations in various formats from CrossRef.*

Description

Get citations in various formats from CrossRef.

Usage

```
cr_cn(
  dois,
  format = "bibtex",
  style = "apa",
  locale = "en-US",
  raw = FALSE,
  .progress = "none",
  url = NULL,
  cache = FALSE,
  ...
)
```

Arguments

dois	Search by a single DOI or many DOIs.
format	Name of the format. One of "rdf-xml", "turtle", "citeproc-json", "citeproc-json-ish", "text", "ris", "bibtex" (default), "crossref-xml", "datacite-xml","bibentry", or "crossref-tdm". The format "citeproc-json-ish" is a format that is not quite proper citeproc-json. Note that the package bibtex is required when `format="bibtex"`; the package is in Suggests so is not required when installing rcrossref
style	a CSL style (for text format only). See `get_styles()` for options. Default: 'apa'. If there's a style that CrossRef doesn't support you'll get a `(500) Internal Server Error`
locale	Language locale. See `?Sys.getlocale`
raw	(logical) Return raw text in the format given by `format` parameter. Default: `FALSE`
.progress	Show a `plyr`-style progress bar? Options are "none", "text", "tk", "win", and "time". See `create_progress_bar` for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the `cursor` param. When using the `cursor` param, this argument only accept a boolean, either `TRUE` or `FALSE`; any non-boolean is coerced to `FALSE`.
url	(character) Base URL for the content negotiation request. Default: "https://doi.org"
cache	(logical) Should requests be cached and/or retrieved from the cache? Note that the cache only persists while the package is loaded.
...	Named parameters passed on to `verb-GET`

Details

See http://citation.crosscite.org/docs.html for more info on the Crossref Content Negotiation API service.

DataCite DOIs: Some values of the `format` parameter won't work with DataCite DOIs, i.e. "citeproc-json", "crossref-xml", "crossref-tdm", "onix-xml".

MEDRA DOIs only work with "rdf-xml", "turtle", "citeproc-json-ish", "ris", "bibtex", "bibentry", "onix-xml".

See examples below.

See cr_agency()

Note that the format type citeproc-json uses the CrossRef API at api.crossref.org, while all others are content negotiated via http://data.crossref.org, http://data.datacite.org or http://data.medra.org. DOI agency is checked first (see cr_agency()).

Examples

```
## Not run:
cr_cn(dois="10.1126/science.169.3946.635")
cr_cn(dois="10.1126/science.169.3946.635", "citeproc-json")
cr_cn(dois="10.1126/science.169.3946.635", "citeproc-json-ish")
cr_cn("10.1126/science.169.3946.635", "rdf-xml")
cr_cn("10.1126/science.169.3946.635", "crossref-xml")
cr_cn("10.1126/science.169.3946.635", "text")

# return an R bibentry type
cr_cn("10.1126/science.169.3946.635", "bibentry")
cr_cn("10.6084/m9.figshare.97218", "bibentry")

# return an apa style citation
cr_cn("10.1126/science.169.3946.635", "text", "apa")
cr_cn("10.1126/science.169.3946.635", "text", "harvard3")
cr_cn("10.1126/science.169.3946.635", "text", "elsevier-harvard")
cr_cn("10.1126/science.169.3946.635", "text", "ecoscience")
cr_cn("10.1126/science.169.3946.635", "text", "heredity")
cr_cn("10.1126/science.169.3946.635", "text", "oikos")

# example with many DOIs
dois <- cr_r(2)
cr_cn(dois, "text", "apa")

# Cycle through random styles - print style on each try
stys <- get_styles()
foo <- function(x){
 cat(sprintf("<Style>:%s\n", x), sep = "\n\n")
 cat(cr_cn("10.1126/science.169.3946.635", "text", style=x))
}
foo(sample(stys, 1))

# Using DataCite DOIs
## some formats don't work
# cr_cn("10.5284/1011335", "crossref-xml")
# cr_cn("10.5284/1011335", "crossref-tdm")
## But most do work
cr_cn("10.5284/1011335", "text")
cr_cn("10.5284/1011335", "datacite-xml")
cr_cn("10.5284/1011335", "rdf-xml")
cr_cn("10.5284/1011335", "turtle")
cr_cn("10.5284/1011335", "citeproc-json-ish")
cr_cn("10.5284/1011335", "ris")
cr_cn("10.5284/1011335", "bibtex")
cr_cn("10.5284/1011335", "bibentry")
```

```
# Using Medra DOIs
cr_cn("10.1430/8105", "onix-xml")

# Get raw output
cr_cn(dois = "10.1002/app.27716", format = "citeproc-json", raw = TRUE)

# sometimes messy DOIs even work
## in this case, a DOI minting agency can't be found
## but we proceed anyway, just assuming it's "crossref"
cr_cn("10.1890/0012-9615(1999)069[0569:EDILSA]2.0.CO;2")

# Use a different base url
cr_cn("10.1126/science.169.3946.635", "text", url = "http://dx.doi.org")
cr_cn("10.1126/science.169.3946.635", "text", "heredity", url = "http://dx.doi.org")
cr_cn("10.5284/1011335", url = "https://citation.crosscite.org/format",
   style = "oikos")
cr_cn("10.5284/1011335", url = "https://citation.crosscite.org/format",
   style = "plant-cell-and-environment")

# A temporary cache can be used to avoid sending the same request repeatedly
# within the same R session.
cr_cn("10.5284/1011335", cache = TRUE)

## End(Not run)
```

`cr_funders` *Search the CrossRef Fundref API*

Description

Search the CrossRef Fundref API

Usage

```
cr_funders(
  dois = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
```

```
  flq = NULL,
  select = NULL,
  ...
)

cr_funders_(
  dois = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  parse = FALSE,
  flq = NULL,
  select = NULL,
  ...
)
```

Arguments

dois	Search by a single DOI or many DOIs.
query	Query terms
filter	Filter options. See examples for usage examples and filters for what filters are available. filter is available for use with cr_works and other crossref family functions with works=TRUE
offset	Number of record to start at. Minimum: 1. For cr_works, and any function setting works = TRUE, the maximum offset value is 10000. For larger requests use cursor.
limit	Number of results to return in the query. Not relavant when searching with specific dois. Default: 20. Max: 1000
sample	(integer) Number of random results to return. when you use the sample parameter, the rows and offset parameters are ignored. Ignored unless works=TRUE. Max: 100
sort	Field to sort on. Acceptable set of fields to sort on: • score OR relevance - Sort by relevance score • updated - Sort by date of most recent change to metadata. Currently the same as deposited. • deposited - Sort by time of most recent deposit • indexed - Sort by time of most recent index

- published - Sort by publication date
- published-print - Sort by print publication date
- published-online - Sort by online publication date
- issued - Sort by issued date (earliest known publication date)
- is-referenced-by-count - Sort by number of times this DOI is referenced by other Crossref DOIs
- references-count - Sort by number of references included in the references section of the document identified by this DOI

order (character) Sort order, one of 'asc' or 'desc'

facet (logical) Include facet results. Boolean or string with field to facet on. Valid fields are *, affiliation, funder-name, funder-doi, orcid, container-title, assertion, archive, update-type, issn, published, source, type-name, publisher-name, license, category-name, assertion-group. Default: FALSE

works (logical) If TRUE, works returned as well, if not then not.

cursor (character) Cursor character string to do deep paging. Default is None. Pass in '*' to start deep paging. Any combination of query, filters and facets may be used with deep paging cursors. While the limit parameter may be specified along with cursor, offset and sample cannot be used with the cursor. See https://github.com/CrossRef/rest-api-doc#deep-paging-with-cursors

cursor_max (integer) Max records to retrieve. Only used when cursor param used. Because deep paging can result in continuous requests until all are retrieved, use this parameter to set a maximum number of records. Of course, if there are less records found than this value, you will get only those found. When cursor pagination is being used the limit parameter sets the chunk size per request.

.progress Show a plyr-style progress bar? Options are "none", "text", "tk", "win", and "time". See create_progress_bar for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the cursor param. When using the cursor param, this argument only accept a boolean, either TRUE or FALSE; any non-boolean is coerced to FALSE.

flq field queries. One or more field queries. Acceptable set of field query parameters are:

- query.container-title - Query container-title aka. publication name
- query.author - Query author first and given names
- query.editor - Query editor first and given names
- query.chair - Query chair first and given names
- query.translator - Query translator first and given names
- query.contributor - Query author, editor, chair and translator first and given names
- query.bibliographic - Query bibliographic information, useful for citation lookup. Includes titles, authors, ISSNs and publication years
- query.affiliation - Query contributor affiliations

Note: query.title has been removed - use query.bibliographic as a replacement

select (character) One or more field to return (only those fields are returned)

...	Named parameters passed on to verb-GET
parse	(logical) Whether to output json FALSE or parse to list TRUE. Default: FALSE

Details

BEWARE: The API will only work for CrossRef DOIs.

This function name changing to cr_funders in the next version - both work for now

Deep paging (using the cursor)

When using the cursor, a character string called next-cursor is returned from Crossref that we use to do the next request, and so on. We use a while loop to get number of results up to the value of cursor_max. Since we are doing each request for you, you may not need the next-cursor string, but if you do want it, you can get to it by indexing into the result like x$meta$next_cursor

Note that you can pass in curl options when using cursor, via "...."

NOTE

Funders without IDs don't show up on the /funders route, and in this function. Some funders don't have assigned IDs in Crossref's system, so won't show up in searches.

Note

See the "Rate limiting" seciton in rcrossref to get into the "fast lane"

References

https://github.com/CrossRef/rest-api-doc

See Also

Other crossref: cr_journals(), cr_licenses(), cr_members(), cr_prefixes(), cr_types(), cr_works()

Examples

```
## Not run:
cr_funders(query="NSF", limit=1)
cr_funders(query="NSF")
cr_funders(dois='10.13039/100000001')
out <- cr_funders(dois=c('10.13039/100000001','10.13039/100000015'))
out['10.13039/100000001']
out[['10.13039/100000001']]

cr_funders(dois='10.13039/100000001')
cr_funders(dois='10.13039/100000001', works=TRUE, limit=5)

cr_funders(dois=c('10.13039/100000001','10.13039/100000015'))
cr_funders(dois=c('10.13039/100000001','10.13039/100000015'), works=TRUE)
```

```
## get facets back
cr_funders("10.13039/100000001", works=TRUE, facet=TRUE, limit = 0)
cr_funders("10.13039/100000001", works=TRUE, facet="license:*", limit = 0)
cr_funders('100000001', works = TRUE, cursor = "*", cursor_max = 500,
   limit = 100, facet=TRUE)

# Curl options
cr_funders(dois='10.13039/100000001', verbose = TRUE)

# If not found, and > 1 DOI given, those not found dropped
cr_funders(dois=c("adfadfaf","asfasf"))
cr_funders(dois=c("adfadfaf","asfasf"), works=TRUE)
cr_funders(dois=c("10.13039/100000001","asfasf"))
cr_funders(dois=c("10.13039/100000001","asfasf"), works=TRUE)

# Use the cursor for deep paging
cr_funders('100000001', works = TRUE, cursor = "*", cursor_max = 500,
   limit = 100)
cr_funders(c('100000001', '100000002'), works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
## with optional progress bar
cr_funders('100000001', works = TRUE, cursor = "*", cursor_max = 500,
   limit = 100, .progress = TRUE)

# Low level function - does no parsing to data.frame, get json or a list
cr_funders_(query = 'nsf')
cr_funders_('10.13039/100000001')
cr_funders_(query = 'science', parse=TRUE)
cr_funders_('10.13039/100000001', works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
cr_funders_('10.13039/100000001', works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100, parse = TRUE)

# field queries
## query.container-title
cr_funders('10.13039/100000001', works = TRUE,
  flq = c(`query.container-title` = 'Ecology'))

# select only certain fields to return
res <- cr_funders('10.13039/100000001', works = TRUE,
  select = c('DOI', 'title'))
names(res$data)

## End(Not run)
```

`cr_journals` *Search CrossRef journals*

Description

Search CrossRef journals

Usage

```
cr_journals(
  issn = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  flq = NULL,
  select = NULL,
  ...
)

cr_journals_(
  issn = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  parse = FALSE,
  flq = NULL,
  select = NULL,
  ...
)
```

Arguments

issn	(character) One or more ISSN's. Format: XXXX-XXXX.
query	Query terms
filter	Filter options. See examples for usage examples and filters for what filters are available. filter is available for use with cr_works and other crossref family functions with works=TRUE

offset — Number of record to start at. Minimum: 1. For cr_works, and any function setting works = TRUE, the maximum offset value is 10000. For larger requests use cursor.

limit — Number of results to return in the query. Not relavant when searching with specific dois. Default: 20. Max: 1000

sample — (integer) Number of random results to return. when you use the sample parameter, the rows and offset parameters are ignored. Ignored unless works=TRUE. Max: 100

sort — Field to sort on. Acceptable set of fields to sort on:

- score OR relevance - Sort by relevance score
- updated - Sort by date of most recent change to metadata. Currently the same as deposited.
- deposited - Sort by time of most recent deposit
- indexed - Sort by time of most recent index
- published - Sort by publication date
- published-print - Sort by print publication date
- published-online - Sort by online publication date
- issued - Sort by issued date (earliest known publication date)
- is-referenced-by-count - Sort by number of times this DOI is referenced by other Crossref DOIs
- references-count - Sort by number of references included in the references section of the document identified by this DOI

order — (character) Sort order, one of 'asc' or 'desc'

facet — (logical) Include facet results. Boolean or string with field to facet on. Valid fields are *, affiliation, funder-name, funder-doi, orcid, container-title, assertion, archive, update-type, issn, published, source, type-name, publisher-name, license, category-name, assertion-group. Default: FALSE

works — (logical) If TRUE, works returned as well, if not then not.

cursor — (character) Cursor character string to do deep paging. Default is None. Pass in '*' to start deep paging. Any combination of query, filters and facets may be used with deep paging cursors. While the limit parameter may be specified along with cursor, offset and sample cannot be used with the cursor. See https://github.com/CrossRef/rest-api-doc#deep-paging-with-cursors

cursor_max — (integer) Max records to retrieve. Only used when cursor param used. Because deep paging can result in continuous requests until all are retrieved, use this parameter to set a maximum number of records. Of course, if there are less records found than this value, you will get only those found. When cursor pagination is being used the limit parameter sets the chunk size per request.

.progress — Show a plyr-style progress bar? Options are "none", "text", "tk", "win", and "time". See create_progress_bar for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the cursor param. When using the cursor param, this argument only accept a boolean, either TRUE or FALSE; any non-boolean is coerced to FALSE.

flq	field queries. One or more field queries. Acceptable set of field query parameters are: • query.container-title - Query container-title aka. publication name • query.author - Query author first and given names • query.editor - Query editor first and given names • query.chair - Query chair first and given names • query.translator - Query translator first and given names • query.contributor - Query author, editor, chair and translator first and given names • query.bibliographic - Query bibliographic information, useful for citation lookup. Includes titles, authors, ISSNs and publication years • query.affiliation - Query contributor affiliations Note: query.title has been removed - use query.bibliographic as a replacement
select	(character) One or more field to return (only those fields are returned)
...	Named parameters passed on to verb-GET
parse	(logical) Whether to output json FALSE or parse to list TRUE. Default: FALSE

Details

BEWARE: The API will only work for CrossRef DOIs.

Note that some parameters are ignored unless works=TRUE: sample, sort, order, filter

Deep paging (using the cursor)

When using the cursor, a character string called next-cursor is returned from Crossref that we use to do the next request, and so on. We use a while loop to get number of results up to the value of cursor_max. Since we are doing each request for you, you may not need the next-cursor string, but if you do want it, you can get to it by indexing into the result like x$meta$next_cursor

Note that you can pass in curl options when using cursor, via "...."

Explanation of some data fields

- backfile_dois: Back file records have a publication date older than two years ago.
- current_dois: Current records are anything published in the last two years.

Note

See the "Rate limiting" seciton in rcrossref to get into the "fast lane"

References

https://github.com/CrossRef/rest-api-doc

See Also

Other crossref: cr_funders(), cr_licenses(), cr_members(), cr_prefixes(), cr_types(), cr_works()

Examples

```
## Not run:
cr_journals(issn="2167-8359")
cr_journals()
cr_journals(issn="2167-8359", works=TRUE)
cr_journals(issn=c('1803-2427','2326-4225'))
cr_journals(query="ecology")
cr_journals(issn="2167-8359", query='ecology', works=TRUE,
   sort='score', order="asc")
cr_journals(issn="2167-8359", query='ecology', works=TRUE, sort='score',
   order="desc")
cr_journals(issn="2167-8359", works=TRUE,
   filter=c(from_pub_date='2014-03-03'))
cr_journals(query="peerj")
cr_journals(issn='1803-2427', works=TRUE)
cr_journals(issn='1803-2427', works=TRUE, sample=1)
cr_journals(limit=2)

## get facets back
cr_journals('1803-2427', works=TRUE, facet=TRUE)
cr_journals('1803-2427', works=TRUE, facet="published:*", limit = 0)
cr_journals(issn=c('1803-2427','2326-4225'), works=TRUE,
  facet="published:*", limit = 10)

# Use the cursor for deep paging
cr_journals(issn='1932-6203', works = TRUE, cursor = "*", cursor_max = 500,
   limit = 100)
cr_journals(c('1932-6203', '0028-0836'), works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
## with optional progress bar
cr_journals(issn='1932-6203', works = TRUE, cursor = "*", cursor_max = 90,
   limit = 30, .progress = TRUE)

# fails, if you want works, you must give an ISSN
# cr_journals(query = "ecology", filter=c(has_full_text = TRUE),
#    works = TRUE)

# Low level function - does no parsing to data.frame, get json or a list
cr_journals_(query = 'ecology')
cr_journals_("2167-8359")
cr_journals_(query = 'ecology', parse=TRUE)
cr_journals_("2167-8359", works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
cr_journals_("2167-8359", works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100, parse = TRUE)

# field queries
```

```
## query.author
cr_journals("2167-8359", works = TRUE, flq = c(`query.author` = 'Jane'))

# select only certain fields to return
res <- cr_journals('2167-8359', works = TRUE,
  select = c('DOI', 'title'))
names(res$data)

## End(Not run)
```

`cr_licenses` *Search CrossRef licenses*

Description

Search CrossRef licenses

Usage

```
cr_licenses(
  query = NULL,
  offset = NULL,
  limit = NULL,
  sort = NULL,
  order = NULL,
  .progress = "none",
  ...
)

cr_licenses_(
  query = NULL,
  offset = NULL,
  limit = NULL,
  sort = NULL,
  order = NULL,
  .progress = "none",
  parse = FALSE,
  ...
)
```

Arguments

`query`	Query terms
`offset`	Number of record to start at, from 1 to infinity.
`limit`	Number of results to return in the query. Not relavant when searching with specific dois. Default: 20. Max: 1000
`sort`	Field to sort on. Acceptable set of fields to sort on:

- score OR relevance - Sort by relevance score
- updated - Sort by date of most recent change to metadata. Currently the same as deposited.
- deposited - Sort by time of most recent deposit
- indexed - Sort by time of most recent index
- published - Sort by publication date
- published-print - Sort by print publication date
- published-online - Sort by online publication date
- issued - Sort by issued date (earliest known publication date)
- is-referenced-by-count - Sort by number of times this DOI is referenced by other Crossref DOIs
- references-count - Sort by number of references included in the references section of the document identified by this DOI

order	(character) Sort order, one of 'asc' or 'desc'
.progress	Show a plyr-style progress bar? Options are "none", "text", "tk", "win, and "time". See plyr::create_progress_bar() for details of each.
...	Named parameters passed on to crul::HttpClient()
parse	(logical) Whether to output json FALSE or parse to list TRUE. Default: FALSE

Details

BEWARE: The API will only work for CrossRef DOIs.

NOTE: The API route behind this function does not support filters any more, so the filter parameter has been removed.

References

https://github.com/CrossRef/rest-api-doc

See Also

Other crossref: cr_funders(), cr_journals(), cr_members(), cr_prefixes(), cr_types(), cr_works()

Examples

```
## Not run:
cr_licenses()
# query for something, e.g. a publisher
cr_licenses(query = 'elsevier')

# Low level function - does no parsing to data.frame, get json or a list
cr_licenses_()
cr_licenses_(query = "elsevier")
cr_licenses_(query = "elsevier", parse=TRUE)

## End(Not run)
```

cr_members *Search CrossRef members*

Description

Search CrossRef members

Usage

```
cr_members(
  member_ids = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  flq = NULL,
  select = NULL,
  ...
)

cr_members_(
  member_ids = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  parse = FALSE,
  flq = NULL,
  select = NULL,
  ...
)
```

Arguments

member_ids	One or more member ids. See examples. Alternatively, you can query for them using the query parameter.
query	Query terms
filter	Filter options. See examples for usage examples and filters for what filters are available. filter is available for use with cr_works and other crossref family functions with works=TRUE
offset	Number of record to start at. Minimum: 1. For cr_works, and any function setting works = TRUE, the maximum offset value is 10000. For larger requests use cursor.
limit	Number of results to return in the query. Not relavant when searching with specific dois. Default: 20. Max: 1000
sample	(integer) Number of random results to return. when you use the sample parameter, the rows and offset parameters are ignored. Ignored unless works=TRUE. Max: 100
sort	Field to sort on. Acceptable set of fields to sort on: • score OR relevance - Sort by relevance score • updated - Sort by date of most recent change to metadata. Currently the same as deposited. • deposited - Sort by time of most recent deposit • indexed - Sort by time of most recent index • published - Sort by publication date • published-print - Sort by print publication date • published-online - Sort by online publication date • issued - Sort by issued date (earliest known publication date) • is-referenced-by-count - Sort by number of times this DOI is referenced by other Crossref DOIs • references-count - Sort by number of references included in the references section of the document identified by this DOI
order	(character) Sort order, one of 'asc' or 'desc'
facet	(logical) Include facet results. Boolean or string with field to facet on. Valid fields are *, affiliation, funder-name, funder-doi, orcid, container-title, assertion, archive, update-type, issn, published, source, type-name, publisher-name, license, category-name, assertion-group. Default: FALSE
works	(logical) If TRUE, works returned as well, if not then not.
cursor	(character) Cursor character string to do deep paging. Default is None. Pass in '*' to start deep paging. Any combination of query, filters and facets may be used with deep paging cursors. While the limit parameter may be specified along with cursor, offset and sample cannot be used with the cursor. See https://github.com/CrossRef/rest-api-doc#deep-paging-with-cursors
cursor_max	(integer) Max records to retrieve. Only used when cursor param used. Because deep paging can result in continuous requests until all are retrieved, use this parameter to set a maximum number of records. Of course, if there are less records

found than this value, you will get only those found. When cursor pagination is being used the `limit` parameter sets the chunk size per request.

`.progress` Show a `plyr`-style progress bar? Options are "none", "text", "tk", "win", and "time". See `create_progress_bar` for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the `cursor` param. When using the `cursor` param, this argument only accept a boolean, either `TRUE` or `FALSE`; any non-boolean is coerced to `FALSE`.

`flq` field queries. One or more field queries. Acceptable set of field query parameters are:

- `query.container-title` - Query container-title aka. publication name
- `query.author` - Query author first and given names
- `query.editor` - Query editor first and given names
- `query.chair` - Query chair first and given names
- `query.translator` - Query translator first and given names
- `query.contributor` - Query author, editor, chair and translator first and given names
- `query.bibliographic` - Query bibliographic information, useful for citation lookup. Includes titles, authors, ISSNs and publication years
- `query.affiliation` - Query contributor affiliations

Note: `query.title` has been removed - use `query.bibliographic` as a replacement

`select` (character) One or more field to return (only those fields are returned)

`...` Named parameters passed on to `verb-GET`

`parse` (logical) Whether to output json `FALSE` or parse to list `TRUE`. Default: `FALSE`

Details

BEWARE: The API will only work for CrossRef DOIs.

Deep paging (using the cursor)

When using the cursor, a character string called `next-cursor` is returned from Crossref that we use to do the next request, and so on. We use a while loop to get number of results up to the value of `cursor_max`. Since we are doing each request for you, you may not need the `next-cursor` string, but if you do want it, you can get to it by indexing into the result like `x$meta$next_cursor`

Note that you can pass in curl options when using cursor, via `"..."`

Note

See the "Rate limiting" seciton in rcrossref to get into the "fast lane"

References

https://github.com/CrossRef/rest-api-doc

See Also

Other crossref: cr_funders(), cr_journals(), cr_licenses(), cr_prefixes(), cr_types(), cr_works()

Examples

```
## Not run:
cr_members(member_ids=98)
cr_members(member_ids=340)

cr_members(member_ids=98, works=TRUE)
cr_members(member_ids=c(10,98,45,1,9))
cr_members(member_ids=c(10,98,45,1,9), works=TRUE)

cr_members(query='hindawi')
cr_members(query='ecology')

# facets
cr_members(member_ids=98, works=TRUE, facet=TRUE, limit = 0)
cr_members(member_ids=98, works=TRUE, facet="license:*", limit = 0)

# curl options
cr_members(member_ids=98, verbose = TRUE)

# Use the cursor for deep paging
cr_members(member_ids=98, works = TRUE, cursor = "*",
   cursor_max = 500, limit = 100)
cr_members(member_ids=c(10, 98, 45), works = TRUE, cursor = "*",
   cursor_max = 200, limit = 100)
## with optional progress bar
cr_members(member_ids=98, works = TRUE, cursor = "*",
   cursor_max = 500, limit = 100, .progress = TRUE)

# data not found
# cr_members(query="adfdf")
# cr_members(member_ids=c(323234343434,3434343434), works=TRUE, facet=TRUE)
# cr_members(member_ids=c(323234343434,3434343434,98), works=TRUE,facet=TRUE)

# Low level function - does no parsing to data.frame, get json or a list
cr_members_(query = 'hindawi')
cr_members_(member_ids = 98)
cr_members_(query = 'hindawi', parse=TRUE)
cr_members_(member_ids = 98, works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
cr_members_(member_ids = 98, works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100, parse=TRUE)

# field queries
## query.container-title
cr_members(98, works = TRUE, flq = c(`query.container-title` = 'Ecology'))
```

```
# select only certain fields to return
res <- cr_members(98, works = TRUE, select = c('DOI', 'title'))
names(res$data)

# Passing a filter parameter as named value to query for index updates
## Using backtick escaped CrossRef documented filter in list of named values
cr_members(1988, works=TRUE, filter=list(`from-index-date`='2023-02-06'))
## Using rcrossref naming convention in named character vector
cr_members(1988, works=TRUE, filter=c(from_index_date='2023-02-06'))

## End(Not run)
```

`cr_prefixes` *Search CrossRef prefixes*

Description

Search CrossRef prefixes

Usage

```
cr_prefixes(
  prefixes,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  flq = NULL,
  select = NULL,
  ...
)

cr_prefixes_(
  prefixes,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
```

```
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  parse = FALSE,
  flq = NULL,
  select = NULL,
  ...
)
```

Arguments

prefixes	(character) Publisher prefixes, one or more in a vector or list. Required.
query	Query terms
filter	Filter options. See examples for usage examples and filters for what filters are available. filter is available for use with cr_works and other crossref family functions with works=TRUE
offset	Number of record to start at. Minimum: 1. For cr_works, and any function setting works = TRUE, the maximum offset value is 10000. For larger requests use cursor.
limit	Number of results to return in the query. Not relavant when searching with specific dois. Default: 20. Max: 1000
sample	(integer) Number of random results to return. when you use the sample parameter, the rows and offset parameters are ignored. Ignored unless works=TRUE. Max: 100
sort	Field to sort on. Acceptable set of fields to sort on: • score OR relevance - Sort by relevance score • updated - Sort by date of most recent change to metadata. Currently the same as deposited. • deposited - Sort by time of most recent deposit • indexed - Sort by time of most recent index • published - Sort by publication date • published-print - Sort by print publication date • published-online - Sort by online publication date • issued - Sort by issued date (earliest known publication date) • is-referenced-by-count - Sort by number of times this DOI is referenced by other Crossref DOIs • references-count - Sort by number of references included in the references section of the document identified by this DOI
order	(character) Sort order, one of 'asc' or 'desc'
facet	(logical) Include facet results. Boolean or string with field to facet on. Valid fields are *, affiliation, funder-name, funder-doi, orcid, container-title, assertion, archive, update-type, issn, published, source, type-name, publisher-name, license, category-name, assertion-group. Default: FALSE

works (logical) If TRUE, works returned as well, if not then not.

cursor (character) Cursor character string to do deep paging. Default is None. Pass in '*' to start deep paging. Any combination of query, filters and facets may be used with deep paging cursors. While the limit parameter may be specified along with cursor, offset and sample cannot be used with the cursor. See https://github.com/CrossRef/rest-api-doc#deep-paging-with-cursors

cursor_max (integer) Max records to retrieve. Only used when cursor param used. Because deep paging can result in continuous requests until all are retrieved, use this parameter to set a maximum number of records. Of course, if there are less records found than this value, you will get only those found. When cursor pagination is being used the limit parameter sets the chunk size per request.

.progress Show a plyr-style progress bar? Options are "none", "text", "tk", "win", and "time". See create_progress_bar for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the cursor param. When using the cursor param, this argument only accept a boolean, either TRUE or FALSE; any non-boolean is coerced to FALSE.

flq field queries. One or more field queries. Acceptable set of field query parameters are:

- query.container-title - Query container-title aka. publication name
- query.author - Query author first and given names
- query.editor - Query editor first and given names
- query.chair - Query chair first and given names
- query.translator - Query translator first and given names
- query.contributor - Query author, editor, chair and translator first and given names
- query.bibliographic - Query bibliographic information, useful for citation lookup. Includes titles, authors, ISSNs and publication years
- query.affiliation - Query contributor affiliations

Note: query.title has been removed - use query.bibliographic as a replacement

select (character) One or more field to return (only those fields are returned)

... Named parameters passed on to verb-GET

parse (logical) Whether to output json FALSE or parse to list TRUE. Default: FALSE

Details

BEWARE: The API will only work for CrossRef DOIs.

Note that any one publisher can have more than one DOI. If you want to search on all DOIs for a publisher, pass in all DOIs, or see cr_members(), and pass in the member_ids parameter.

Notes from CrossRef (quoting them):

The prefix of a CrossRef DOI does NOT indicate who currently owns the DOI. It only reflects who originally registered the DOI. CrossRef metadata has an owner_prefix element that records the current owner of the CrossRef DOI in question.

CrossRef also has member IDs for depositing organisations. A single member may control multiple owner prefixes, which in turn may control a number of DOIs. When looking at works published by a certain organisation, member IDs and the member routes should be used.

Deep paging (using the cursor)

When using the cursor, a character string called `next-cursor` is returned from Crossref that we use to do the next request, and so on. We use a while loop to get number of results up to the value of `cursor_max`. Since we are doing each request for you, you may not need the `next-cursor` string, but if you do want it, you can get to it by indexing into the result like `x$meta$next_cursor`

Note that you can pass in curl options when using cursor, via `"..."`

Note

See the "Rate limiting" seciton in rcrossref to get into the "fast lane"

References

https://github.com/CrossRef/rest-api-doc

See Also

Other crossref: `cr_funders()`, `cr_journals()`, `cr_licenses()`, `cr_members()`, `cr_types()`, `cr_works()`

Examples

```
## Not run:
cr_prefixes(prefixes="10.1016")
cr_prefixes(prefixes="10.1016", works=TRUE)
cr_prefixes(prefixes=c('10.1016','10.1371','10.1023','10.4176','10.1093'))
cr_prefixes(prefixes=c('10.1016','10.1371'), works=TRUE)
cr_prefixes(prefixes="10.1016", works=TRUE, filter=c(has_full_text=TRUE),
  limit=5)
cr_prefixes(prefixes="10.1016", works=TRUE, query='ecology', limit=4)
cr_prefixes(prefixes="10.1016", works=TRUE, query='ecology', limit=4)

# facets - only avail. when works=TRUE
cr_prefixes(prefixes="10.1016", works=TRUE, facet=TRUE)
cr_prefixes(prefixes="10.1016", works=TRUE, facet="license:*", limit=0)
cr_prefixes(prefixes=c('10.1016','10.1371'), works=TRUE, facet=TRUE,
  limit=0)

# Use the cursor for deep paging
cr_prefixes("10.1016", works = TRUE, cursor = "*", cursor_max = 500,
   limit = 100)
cr_prefixes(c('10.1016', '10.1371'), works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
## with optional progress bar
cr_prefixes("10.1016", works = TRUE, cursor = "*", cursor_max = 500,
   limit = 100, .progress = TRUE)

# Low level function - does no parsing to data.frame, get json or a list
cr_prefixes_("10.1016")
cr_prefixes_(c('10.1016', '10.1371'))
cr_prefixes_("10.1016", works = TRUE, query = 'ecology', limit = 10)
```

```
cr_prefixes_("10.1016", works = TRUE, query = 'ecology', parse=TRUE,
   limit = 10)
cr_prefixes_("10.1016", works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
cr_prefixes_("10.1016", works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100, parse = TRUE)

# field queries
## query.container-title
cr_prefixes("10.1016", works = TRUE,
  flq = c(`query.container-title` = 'Ecology'))

# select only certain fields to return
res <- cr_prefixes("10.1016", works = TRUE, select = c('DOI', 'title'))
names(res$data)

## End(Not run)
```

`cr_r` *Get a random set of DOI's through CrossRef.*

Description

Get a random set of DOI's through CrossRef.

Usage

```
cr_r(sample = 10, ...)
```

Arguments

`sample`	The number of returned random DOIs. Maximum: 100. Default: 20.
`...`	Further args passed on to `cr_works()`

Value

A character vector of DOIs

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
# Default search gets 10 random DOIs
cr_r()

# Get 30 DOIs
```

```
cr_r(30)

## End(Not run)
```

`cr_types` *Search CrossRef types*

Description

Search CrossRef types

Usage

```
cr_types(
  types = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  flq = NULL,
  select = NULL,
  ...
)

cr_types_(
  types = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  works = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  parse = FALSE,
```

```
  flq = NULL,
  select = NULL,
  ...
)
```

Arguments

types	(character) Type identifier, e.g., journal
query	Query terms
filter	Filter options. See examples for usage examples and filters for what filters are available. filter is available for use with cr_works and other crossref family functions with works=TRUE
offset	Number of record to start at. Minimum: 1. For cr_works, and any function setting works = TRUE, the maximum offset value is 10000. For larger requests use cursor.
limit	Number of results to return in the query. Not relavant when searching with specific dois. Default: 20. Max: 1000
sample	(integer) Number of random results to return. when you use the sample parameter, the rows and offset parameters are ignored. Ignored unless works=TRUE. Max: 100
sort	Field to sort on. Acceptable set of fields to sort on: • score OR relevance - Sort by relevance score • updated - Sort by date of most recent change to metadata. Currently the same as deposited. • deposited - Sort by time of most recent deposit • indexed - Sort by time of most recent index • published - Sort by publication date • published-print - Sort by print publication date • published-online - Sort by online publication date • issued - Sort by issued date (earliest known publication date) • is-referenced-by-count - Sort by number of times this DOI is referenced by other Crossref DOIs • references-count - Sort by number of references included in the references section of the document identified by this DOI
order	(character) Sort order, one of 'asc' or 'desc'
facet	(logical) Include facet results. Boolean or string with field to facet on. Valid fields are *, affiliation, funder-name, funder-doi, orcid, container-title, assertion, archive, update-type, issn, published, source, type-name, publisher-name, license, category-name, assertion-group. Default: FALSE
works	(logical) If TRUE, works returned as well, if not then not.
cursor	(character) Cursor character string to do deep paging. Default is None. Pass in '*' to start deep paging. Any combination of query, filters and facets may be used with deep paging cursors. While the limit parameter may be specified along with cursor, offset and sample cannot be used with the cursor. See https://github.com/CrossRef/rest-api-doc#deep-paging-with-cursors

cursor_max	(integer) Max records to retrieve. Only used when cursor param used. Because deep paging can result in continuous requests until all are retrieved, use this parameter to set a maximum number of records. Of course, if there are less records found than this value, you will get only those found. When cursor pagination is being used the `limit` parameter sets the chunk size per request.
.progress	Show a plyr-style progress bar? Options are "none", "text", "tk", "win", and "time". See create_progress_bar for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the `cursor` param. When using the `cursor` param, this argument only accept a boolean, either `TRUE` or `FALSE`; any non-boolean is coerced to `FALSE`.
flq	field queries. One or more field queries. Acceptable set of field query parameters are: • `query.container-title` - Query container-title aka. publication name • `query.author` - Query author first and given names • `query.editor` - Query editor first and given names • `query.chair` - Query chair first and given names • `query.translator` - Query translator first and given names • `query.contributor` - Query author, editor, chair and translator first and given names • `query.bibliographic` - Query bibliographic information, useful for citation lookup. Includes titles, authors, ISSNs and publication years • `query.affiliation` - Query contributor affiliations Note: `query.title` has been removed - use `query.bibliographic` as a replacement
select	(character) One or more field to return (only those fields are returned)
...	Named parameters passed on to verb-GET
parse	(logical) Whether to output json `FALSE` or parse to list `TRUE`. Default: `FALSE`

Details

BEWARE: The API will only work for CrossRef DOIs.

Deep paging (using the cursor)

When using the cursor, a character string called `next-cursor` is returned from Crossref that we use to do the next request, and so on. We use a while loop to get number of results up to the value of `cursor_max`. Since we are doing each request for you, you may not need the `next-cursor` string, but if you do want it, you can get to it by indexing into the result like `x$meta$next_cursor`

Note that you can pass in curl options when using cursor, via `"..."`

Note

See the "Rate limiting" seciton in rcrossref to get into the "fast lane"

References

https://github.com/CrossRef/rest-api-doc

See Also

Other crossref: cr_funders(), cr_journals(), cr_licenses(), cr_members(), cr_prefixes(), cr_works()

Examples

```
## Not run:
cr_types()
cr_types("monograph")
cr_types("monograph", works=TRUE)
cr_types(c('monograph', 'book-set', 'book', 'book-track'))
cr_types(c('monograph', 'book-set'), works=TRUE)

## get facets back
cr_types("journal-article", works=TRUE, facet=TRUE)$facets
cr_types("monograph", works=TRUE, facet="license:*", limit = 0)
cr_types(c('monograph', 'book-set'), works=TRUE, facet=TRUE)

# Use the cursor for deep paging
cr_types("journal-article", works = TRUE, cursor = "*",
   cursor_max = 500, limit = 100)
cr_types(c('monograph', 'book-set'), works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)
## with optional progress bar
cr_types("journal-article", works = TRUE, cursor = "*",
   cursor_max = 500, limit = 100, .progress = TRUE)

# query doesn't work unless using works=TRUE
### you get results, but query param is silently dropped
cr_types(query = "ecology")

# print progress - only works when passing more than one type
cr_types(c('monograph', 'book-set'), works=TRUE, .progress='text')

# Low level function - does no parsing to data.frame, get json or a list
cr_types_('monograph')
cr_types_('monograph', parse = TRUE)
cr_types_("journal-article", works = TRUE, cursor = "*",
   cursor_max = 300, limit = 100)

# field queries
## query.container-title
cr_types("journal-article", works = TRUE,
  flq = c(`query.container-title` = 'Ecology'))

# select only certain fields to return
res <- cr_types("journal-article", works = TRUE, select = c('DOI', 'title'))
names(res$data)

## End(Not run)
```

cr_works *Search CrossRef works (articles)*

Description

Search CrossRef works (articles)

Usage

```
cr_works(
  dois = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  flq = NULL,
  select = NULL,
  async = FALSE,
  ...
)

cr_works_(
  dois = NULL,
  query = NULL,
  filter = NULL,
  offset = NULL,
  limit = NULL,
  sample = NULL,
  sort = NULL,
  order = NULL,
  facet = FALSE,
  cursor = NULL,
  cursor_max = 5000,
  .progress = "none",
  parse = FALSE,
  flq = NULL,
  select = NULL,
  async = FALSE,
  ...
)
```

Arguments

dois	Search by a single DOI or many DOIs. Note that using this parameter at the same time as the query, limit, select or flq parameter will result in an error.
query	Query terms
filter	Filter options. See examples for usage examples and filters for what filters are available. filter is available for use with cr_works and other crossref family functions with works=TRUE
offset	Number of record to start at. Minimum: 1. For cr_works, and any function setting works = TRUE, the maximum offset value is 10000. For larger requests use cursor.
limit	Number of results to return in the query. Not relavant when searching with specific dois. Default: 20. Max: 1000
sample	(integer) Number of random results to return. when you use the sample parameter, the rows and offset parameters are ignored. Ignored unless works=TRUE. Max: 100
sort	Field to sort on. Acceptable set of fields to sort on: • score OR relevance - Sort by relevance score • updated - Sort by date of most recent change to metadata. Currently the same as deposited. • deposited - Sort by time of most recent deposit • indexed - Sort by time of most recent index • published - Sort by publication date • published-print - Sort by print publication date • published-online - Sort by online publication date • issued - Sort by issued date (earliest known publication date) • is-referenced-by-count - Sort by number of times this DOI is referenced by other Crossref DOIs • references-count - Sort by number of references included in the references section of the document identified by this DOI
order	(character) Sort order, one of 'asc' or 'desc'
facet	(logical) Include facet results. Boolean or string with field to facet on. Valid fields are *, affiliation, funder-name, funder-doi, orcid, container-title, assertion, archive, update-type, issn, published, source, type-name, publisher-name, license, category-name, assertion-group. Default: FALSE
cursor	(character) Cursor character string to do deep paging. Default is None. Pass in '*' to start deep paging. Any combination of query, filters and facets may be used with deep paging cursors. While the limit parameter may be specified along with cursor, offset and sample cannot be used with the cursor. See https://github.com/CrossRef/rest-api-doc#deep-paging-with-cursors
cursor_max	(integer) Max records to retrieve. Only used when cursor param used. Because deep paging can result in continuous requests until all are retrieved, use this parameter to set a maximum number of records. Of course, if there are less records found than this value, you will get only those found. When cursor pagination is being used the limit parameter sets the chunk size per request.

.progress	Show a plyr-style progress bar? Options are "none", "text", "tk", "win", and "time". See create_progress_bar for details of each. Only used when passing in multiple ids (e.g., multiple DOIs, DOI prefixes, etc.), or when using the cursor param. When using the cursor param, this argument only accept a boolean, either TRUE or FALSE; any non-boolean is coerced to FALSE.
flq	field queries. One or more field queries. Acceptable set of field query parameters are: • query.container-title - Query container-title aka. publication name • query.author - Query author first and given names • query.editor - Query editor first and given names • query.chair - Query chair first and given names • query.translator - Query translator first and given names • query.contributor - Query author, editor, chair and translator first and given names • query.bibliographic - Query bibliographic information, useful for citation lookup. Includes titles, authors, ISSNs and publication years • query.affiliation - Query contributor affiliations Note: query.title has been removed - use query.bibliographic as a replacement
select	(character) One or more field to return (only those fields are returned)
async	(logical) use async HTTP requests. Default: FALSE
...	Named parameters passed on to verb-GET
parse	(logical) Whether to output json FALSE or parse to list TRUE. Default: FALSE

Deep paging (using the cursor)

When using the cursor, a character string called next-cursor is returned from Crossref that we use to do the next request, and so on. We use a while loop to get number of results up to the value of cursor_max. Since we are doing each request for you, you may not need the next-cursor string, but if you do want it, you can get to it by indexing into the result like x$meta$next_cursor

Note that you can pass in curl options when using cursor, via "..."

Beware

The API will only work for CrossRef DOIs.

Functions

- cr_works() - Does data request and parses to data.frame for easy downstream consumption
- cr_works_() - Does data request, and gives back json (default) or lists, with no attempt to parse to data.frame's

Explanation of some data fields

- score: a term frequency, inverse document frequency score that comes from the Crossref Solr backend, based on bibliographic metadata fields title, publication title, authors, ISSN, publisher, and date of publication.

Note

See the "Rate limiting" seciton in rcrossref to get into the "fast lane"

References

https://github.com/CrossRef/rest-api-doc

See Also

Other crossref: cr_funders(), cr_journals(), cr_licenses(), cr_members(), cr_prefixes(), cr_types()

Examples

```
## Not run:
# Works funded by the NSF
cr_works(query="NSF")

# Works that include renear but not ontologies
cr_works(query="renear+-ontologies")

# Filter
cr_works(query="global state", filter=c(has_orcid=TRUE), limit=3)
# Filter by multiple fields
cr_works(filter=c(has_orcid=TRUE, from_pub_date='2004-04-04'))
# Only full text articles
cr_works(filter=c(has_full_text = TRUE))
# has affilitation data
cr_works(filter=c(has_affiliation = TRUE))
# has abstract
cr_works(filter=c(has_abstract = TRUE))
# has clinical trial number
cr_works(filter=c(has_clinical_trial_number = TRUE))

# Querying dois
cr_works(dois='10.1063/1.3593378')
cr_works('10.1371/journal.pone.0033693')
cr_works(dois='10.1007/12080.1874-1746')
cr_works(dois=c('10.1007/12080.1874-1746','10.1007/10452.1573-5125',
   '10.1111/(issn)1442-9993'))

# progress bar
cr_works(dois=c('10.1007/12080.1874-1746','10.1007/10452.1573-5125'),
   .progress="text")

# Include facetting in results
cr_works(query="NSF", facet=TRUE)
## Get facets only, by setting limit=0
cr_works(query="NSF", facet=TRUE, limit=0)
## you can also set facet to a query
cr_works(facet = "license:*", limit=0)
```

```
# Sort results
cr_works(query="ecology", sort='relevance', order="asc")
res <- cr_works(query="ecology", sort='score', order="asc")
res$data$score
cr_works(query="ecology", sort='published')
x=cr_works(query="ecology", sort='published-print')
x=cr_works(query="ecology", sort='published-online')

# Get a random number of results
cr_works(sample=1)
cr_works(sample=10)

# You can pass in dot separated fields to filter on specific fields
cr_works(filter=c(award.number='CBET-0756451',
   award.funder='10.13039/100000001'))

# Use the cursor for deep paging
cr_works(query="NSF", cursor = "*", cursor_max = 300, limit = 100)
cr_works(query="NSF", cursor = "*", cursor_max = 300, limit = 100,
   facet = TRUE)
## with optional progress bar
x <- cr_works(query="NSF", cursor = "*", cursor_max = 1200, limit = 200,
  .progress = TRUE)

# Low level function - does no parsing to data.frame, get json or a list
cr_works_(query = "NSF")
cr_works_(query = "NSF", parse=TRUE)
cr_works_(query="NSF", cursor = "*", cursor_max = 300, limit = 100)
cr_works_(query="NSF", cursor = "*", cursor_max = 300, limit = 100,
   parse=TRUE)

# field queries
## query.author
res <- cr_works(query = "ecology", flq = c(query.author = 'Boettiger'))

## query.container-title
res <- cr_works(query = "ecology",
  flq = c(`query.container-title` = 'Ecology'))

## query.author and query.bibliographic
res <- cr_works(query = "ecology",
  flq = c(query.author = 'Smith', query.bibliographic = 'cell'))

# select only certain fields to return
res <- cr_works(query = "NSF", select = c('DOI', 'title'))
names(res$data)

# asyc
queries <- c("ecology", "science", "cellular", "birds", "European",
  "bears", "beets", "laughter", "hapiness", "funding")
res <- cr_works(query = queries, async = TRUE)
res_json <- cr_works_(query = queries, async = TRUE)
unname(vapply(res_json, class, ""))
```

```
jsonlite::fromJSON(res_json[[1]])

queries <- c("ecology", "science", "cellular")
res <- cr_works(query = queries, async = TRUE, verbose = TRUE)
res

# time
queries <- c("ecology", "science", "cellular", "birds", "European",
  "bears", "beets", "laughter", "hapiness", "funding")
system.time(cr_works(query = queries, async = TRUE))
system.time(lapply(queries, function(z) cr_works(query = z)))

## End(Not run)
```

`filters` *Get filter details and names.*

Description

Get filter details and names.

Usage

```
filter_names()

filter_details(x = NULL)
```

Arguments

x	(character) Optional filter name. If not given, all filters returned.

Details

Note that all filter names in this package have periods and dashes replaced with underscores as they both cause problems in an R console.

Examples

```
filter_names()
filter_details()
filter_details()$has_authenticated_orcid
filter_details()$has_authenticated_orcid$possible_values
filter_details()$has_authenticated_orcid$description
filter_details("issn")
filter_details("iss")
filter_details(c("issn", "alternative_id"))
```

get_styles *Get list of styles from github.com/citation-style-language/styles*

Description

Get list of styles from github.com/citation-style-language/styles

Usage

```
get_styles(...)
```

Arguments

...	Named parameters passed on to crul::HttpClient

Examples

```
## Not run:
x <- get_styles()
x[1:10]

## End(Not run)
```

id_converter *Get a PMID from a DOI, and vice versa.*

Description

Get a PMID from a DOI, and vice versa.

Usage

```
id_converter(x, type = NULL, ...)
```

Arguments

x	(character) One or more of: doi, pmid, pmcid, or manuscript id, see examples. required.
type	(character) one of doi, pmid, pmcid, or manuscript id
...	Curl args passed on to crul::verb-GET

References

https://www.ncbi.nlm.nih.gov/pmc/tools/id-converter-api/

Examples

```
## Not run:
# get a pmid/pmcid from a doi
id_converter("10.1038/ng.590")

# pmid to doi/pmcid
id_converter("20495566", "pmid")
id_converter("20495566")
# id_converter("20495566", "doi") #error

# pmcid to doi/pmid
id_converter("PMC2883744", "pmcid")
id_converter("PMC2883744")

# manuscript id
id_converter("NIHMS311352")

# more than 1 ID
id_converter(c("PMC3531190","PMC3245039"))

# error, wrong type passed for id given
# id_converter("PMC2883744", "doi")
# error, 200 ids or less
# ids <- cr_r(100)
# id_converter(c(ids, ids, ids))

## End(Not run)
```

`rcrossref-defunct` *Defunct functions in rcrossref*

Description

These functions are gone, no longer available.

Details

- `cr_citation()`: Crossref is trying to sunset their OpenURL API, which this function uses. So this function is now removed. See the function `cr_cn()`, which does the same things, but with more functionality, using the new Crossref API.
- `pmid2doi()` and `doi2pmid()`: The API behind these functions is down for good, see `id_converter()` for similar functionality.
- `cr_search()`: The functionality of this function can be achieved with the new Crossref API. See functions `cr_works()` et al.
- `cr_search_free()`: The functionality of this function can be achieved with the new Crossref API. See functions `cr_works()` et al.
- `crosscite()`: The functionality of this function can be achieved with `cr_cn()`

- cr_fundref(): Crossref changed their name "fundref" to "funders", so we've changed our function, see cr_funders()
- cr_ft_text(): This function and other text mining functions are incorporated in a new package crminer.
- cr_ft_links(): This function and other text mining functions are incorporated in a new package crminer.
- cr_ft_pdf(): This function and other text mining functions are incorporated in a new package crminer.
- cr_ft_plain(): This function and other text mining functions are incorporated in a new package crminer.
- cr_ft_text(): This function and other text mining functions are incorporated in a new package crminer.
- cr_ft_xml(): This function and other text mining functions are incorporated in a new package crminer.
- as.tdmurl(): This function and other text mining functions are incorporated in a new package crminer.
- extract_xpdf(): This function and other text mining functions are incorporated in a new package crminer.

---

rcrossref-deprecated *Deprecated functions in rcrossref*

Description

None at the moment

Index