Package 'rdpla'

August 13, 2017

Type Package

Title Client for the Digital Public Library of America ('DPLA')

Description Interact with the Digital Public Library of America <https://dp.la> ('DPLA') 'REST' 'API' <https://dp.la/info/developers/codex/> from R, including search and more.

Version 0.2.0

License MIT + file LICENSE

URL https://github.com/ropensci/rdpla

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

LazyData yes

VignetteBuilder knitr

Imports crul (>= 0.3.8), jsonlite (>= 1.0), tibble, data.table, hoardr

Suggests roxygen2 (>= 6.0.1), knitr, testthat, ggplot2, lubridate, scales

RoxygenNote 6.0.1

NeedsCompilation no

Author Scott Chamberlain [aut, cre]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2017-08-13 17:14:17 UTC

R topics documented:

`rdpla-package` *R Client for the Digital Public Library of America (DPLA).*

Description

Interact with the Digital Public Library of America (DPLA) REST API from R, including search.

For an introduction to **rdpla**, see the vignette **Introduction to rdpla**

Package API

The following are the main functions in **rdpla**

- dpla_items - Work with the DPLA items endpoint
- dpla_collections - Work with the DPLA collections endpoint
- dpla_bulk - Download bulk and compressed JSON data

Authentication

See `dpla_get_key()` for authentication help.

Author(s)

Scott Chamberlain

`country_codes` *Country codes*

Description

A data.frame (249 rows, 2 columns) containing all language codes. The columns are as follows:

Details

- code. country code
- name. country name

`dpla_bulk` *Get bulk DPLA data*

Description

Get bulk DPLA data

Usage

```
dpla_bulk(year, month, key, ...)

dpla_bulk_list(...)
```

Arguments

`year`	(character) a year between 2015 and the current year
`month`	(character) between 1 (January) and 12 (December)
`key`	(character) a dataset name key, see Details.
`...`	Curl options passed on to `crul::HttpClient()`

Details

All data in the DPLA repository are available for download as gzipped JSON files. These include the standard DPLA fields, as well as the complete record received from the partner.

See https://digitalpubliclibraryofamerica.atlassian.net/wiki/spaces/TECH/pages/5931056/Database+export+files for description of the structure of the files

`dpla_bulk` doesn't attempt to read in the bulk JSON files as they can be quite large - so we leave that to the user.

Value

`dpla_bulk_list` returns the partial paths for JSON dataset dumps; append the base URL `https://dpla-provider-export.` to the beginning to get the full URL. `dpla_bulk` returns a path to the compressed JSON file.

Allowed Keys

- all
- artstor
- bhl
- cdl
- david_rumsey
- digital_commonwealth
- digitalnc
- esdn

- georgia
- getty
- gpo
- harvard
- hathitrust
- il
- indiana
- internet_archive
- kdl
- lc
- maine
- maryland
- mdl
- michigan
- missouri_hub
- mwdl
- nara
- nypl
- pa
- pennsylvania
- scdl
- smithsonian
- tennessee
- the_portal_to_texas_history
- tn
- uiuc
- usc
- virginia
- washington
- wisconsin

Examples

```
## Not run:
dpla_bulk(year = 2016, month = 4, key = "nypl")
res <- dpla_bulk(year = 2017, month = 1, key = "artstor")

## End(Not run)
```

dpla_cache *Caching*

Description

Manage cached rdpla files with **hoardr**

Details

The dafault cache directory is `paste0(rappdirs::user_cache_dir(), "/R/rdpla")`, but you can set your own path using `cache_path_set()`

`cache_delete` only accepts 1 file name, while `cache_delete_all` doesn't accept any names, but deletes all files. For deleting many specific files, use `cache_delete` in a `lapply()` type call

Useful user functions

- `dpla_cache$cache_path_get()` get cache path
- `dpla_cache$cache_path_set()` set cache path
- `dpla_cache$list()` returns a character vector of full path file names
- `dpla_cache$files()` returns file objects with metadata
- `dpla_cache$details()` returns files with details
- `dpla_cache$delete()` delete specific files
- `dpla_cache$delete_all()` delete all files, returns nothing

Examples

```
## Not run:
dpla_cache

# list files in cache
dpla_cache$list()

# delete certain database files
# dpla_cache$delete("file path")
# dpla_cache$list()

# delete all files in cache
# dpla_cache$delete_all()
# dpla_cache$list()

# set a different cache path from the default

## End(Not run)
```

`dpla_collections` *Search collections from the Digital Public Library of America (DPLA).*

Description

Search collections from the Digital Public Library of America (DPLA).

Usage

```
dpla_collections(q = NULL, title = NULL, description = NULL,
  fields = NULL, sort_by = NULL, page_size = 10, page = NULL,
  key = NULL, ...)
```

Arguments

`q`	(character) Query terms.
`title`	(character) Query in the title field
`description`	(character) Query in the description field
`fields`	(character) A vector of the fields to return in the output. The default is all fields. See dpla_fields for options.
`sort_by`	(character) The default sort order is ascending. Most, but not all fields can be sorted on. Attempts to sort on an un-sortable field will return the standard error structure with a HTTP 400 status code.
`page_size`	(integer) Number of items to return. Default: 10. Max: 500.
`page`	(integer) Page number to return. Default: `NULL` (which means this parameter is not passed to DPLA, so they default to **1**)
`key`	(character) Your DPLA API key. See `dpla_get_key()`
`...`	Curl options passed on to `crul::HttpClient()`

Value

A list with two slots:

- meta - a tibble/data.frame of metadata for the response, with one row and two columns:
  - found - number of records found matching criteria
  - returned - number of records returned
- data - a tibble/data.frame of the data; empty data.frame if no data matches your request; structure varies depending on search criteria

Examples

```
## Not run:
dpla_collections(q="university of texas", page_size=2)
dpla_collections(q="university of texas", fields='id', page_size=2)
dpla_collections(q="university of texas", sort_by='title', page_size=5)
dpla_collections(title="paso")
dpla_collections(description="east")

# use curl options
dpla_collections(q="university", verbose = TRUE)

## End(Not run)
```

`dpla_fields` *Metadata providers data.frame*

Description

A data.frame (46 rows, 2 columns) containing all the DPLA fields. The columns are as follows:

Details

- field. Name of the field.
- field_description. Description of the field.

`dpla_get_key` *Request an API key from the Digital Public Library of America (DPLA).*

Description

Request an API key from the Digital Public Library of America (DPLA).

Usage

```
dpla_get_key(email, ...)
```

Arguments

`email`	(character) An email address.
`...`	Curl options passed on to `crul::HttpClient()`

Details

You are required to have an API key to use **rdpla**. To get one, use `dpla_get_key` for getting a key programatically. After getting the key, you can pass the key as a parameter to **rdpla** functions, but we recommend storing the key on your machine, since not exposing your key in your files that may end up on the web is good practice. Store your key either as an environment variable in your `.Renviron` file or similar like `DPLA_API_KEY=<yourkey>`, or as an R option in your `.Rprofile` file like `options(dpla_api_key = "<yourkey>")`. Either will be read in when you call **rdpla** functions. Make sure to restart your R session after storing your key as either env var or R option.

Value

On success, a message that your API key will be emailed to you.

Examples

```
## Not run:
# dpla_get_key(email="stuff@thing.com")

## End(Not run)
```

---

dpla_items *Search items from the Digital Public Library of America (DPLA).*

Description

Search items from the Digital Public Library of America (DPLA).

Usage

```
dpla_items(q = NULL, description = NULL, title = NULL, subject = NULL,
  creator = NULL, type = NULL, publisher = NULL, format = NULL,
  rights = NULL, contributor = NULL, provider = NULL, sp = NULL,
  sp_coordinates = NULL, sp_city = NULL, sp_county = NULL,
  sp_distance = NULL, sp_country = NULL, sp_code = NULL, sp_name = NULL,
  sp_region = NULL, sp_state = NULL, fields = NULL, sort_by = NULL,
  date = NULL, date_before = NULL, date_after = NULL, language = NULL,
  page_size = 100, page = NULL, facets = NULL, facet_size = 100,
  key = NULL, what = "table", ...)
```

Arguments

`q`	(character) Query terms.
`description`	(character) Object description.
`title`	(character) Object title.
`subject`	(character) Subject area
`creator`	(character) Creator name

type	(character) Type of object
publisher	(character) Publisher
format	(character) Format
rights	(character) Rights
contributor	(character) Contributor
provider	(character) Provider
sp	(character) Query all spatial fields.
sp_coordinates	(character) Query only coordinates. Of the form <latitude,longitude>
sp_city	(character) Query by city name.
sp_county	(character) Query by county name.
sp_distance	(character) Distance from point defined in the sp_coordinates field
sp_country	(character) Query by location country
sp_code	(character) Query by ISO 3166-2 country code. Codes are included in this package, see country_codes. Find out more at https://www.iso.org/obp/ui/#search
sp_name	(character) Location name.
sp_region	(character) Name of a region, e.g., "Upstate New York" (literal)
sp_state	(character) ISO 3166-2 code for a U.S. state or territory
fields	(character) A vector of the fields to return in the output. The default is all fields. See dpla_fields for options.
sort_by	(character) The default sort order is ascending. Most, but not all fields can be sorted on. Attempts to sort on an un-sortable field will return the standard error structure with a HTTP 400 status code.
date	(character) A date
date_before	(character) Date before
date_after	(character) Date after
language	(character) One of a name of a language, or an ISO-639 code. Codes are included in this package, see language_codes. Find out more at http://www-01.sil.org/iso639-3/default.asp
page_size	(integer) Number of items to return, defaults to 100. Max of 500.
page	(integer) Page number to return, defaults to NULL.
facets	(character) Fields to facet on.
facet_size	(integer) Default to 100, maximum 2000.
key	(character) Your DPLA API key. See dpla_get_key()
what	(character) One of list or table (data.frame). (Default: table)
...	Curl options passed on to crul::HttpClient()

Details

Note that parsing of results right now can lead to multiple rows per record because sometimes an array of length > 1 for a result makes a data.frame of more than one row per record. Thus, you will get duplicated values in the id column of the results.

Value

A list of length three:

- meta - metadata for the call, with one row and three columns:
  - found - number of records found matching criteria
  - start - offset from record 1
  - returned - number of records returned
- data - tibble (data.frame) of results
- facets - list of same length as number of facets requested, each with a list of length two with a meta tibble/data.frame, and a data tibble/data.frame

Examples

```
## Not run:
# Basic search, "fruit" in any fields
dpla_items(q="fruit")

# Limit records returned
dpla_items(q="fruit", page_size=2)

# Return certain fields
dpla_items(q="fruit", fields=c("id","publisher","format"))
dpla_items(q="fruit", fields="subject")

# Max is 500 per call, but you can use combo of page_size and page params
dpla_items(q="fruit", fields="id", page_size=500)$meta$returned
lapply(1:2, function(x) {
   dpla_items(q="fruit", fields="id", page_size=500, page=x)$meta$returned
})
out <- lapply(1:2, function(x) dpla_items(q="fruit", fields="id",
   page_size=500, page=x))
lapply(out, function(y) head(y$data))

# Search by date
out <- dpla_items(q="science", date_before=1900, page_size=200)
out$data

# Search by various fields
dpla_items(description="obituaries", page_size=2, fields="description")
dpla_items(title="obituaries", page_size=2, fields="title")
dpla_items(subject="yodeling", page_size=2, fields="subject")
dpla_items(creator="Holst-Van der Schalk", page_size=2, fields="creator")
dpla_items(type="text", page_size=2, fields="type")
dpla_items(publisher="Leningrad", page_size=2, fields="publisher")
dpla_items(rights="unrestricted", page_size=2, fields="rights")
dpla_items(provider="HathiTrust", page_size=2, fields="provider")

## don't seem to work
# dpla_items(contributor="Smithsonian", page_size=2, fields="contributor")
# dpla_items(format="Electronic resource", page_size=2, fields="format")
```

```
# Spatial search
## sp searches all spatial fields, or search on specific fields, see those
## params with sp_*
dpla_items(sp='Boston', page_size=2)
dpla_items(sp_state='Hawaii', page_size=2)
dpla_items(sp_state='Massachusetts OR Hawaii', page_size=2)
dpla_items(sp_coordinates='40,-100', page_size=2)
dpla_items(sp_country='Canada', page_size=2)
dpla_items(sp_county='Sacramento', page_size=2)

# Language search
dpla_items(language='Russian')$meta
dpla_items(language='rus')$meta
dpla_items(language='English')$meta

# Sorting
dpla_items(fields=c("id","title"), page_size=10)
dpla_items(fields=c("id","title"), page_size=10,
  sort_by="sourceResource.title")

# Faceting
dpla_items(facets="sourceResource.format", page_size=0)
dpla_items(facets="sourceResource.format", page_size=0, facet_size=5)
ids <- c("sourceResource.spatial.state","sourceResource.spatial.country")
dpla_items(facets=ids, page_size=0)
dpla_items(facets="sourceResource.type", page_size=0)
#dpla_items(facets="sourceResource.spatial.coordinates:42.3:-71",
#   page_size=0)
#dpla_items(facets="sourceResource.temporal.begin", page_size=0)
dpla_items(facets="provider.name", page_size=0)
dpla_items(facets="isPartOf", page_size=0)
dpla_items(facets="hasView", page_size=0)

## End(Not run)
```

`language_codes` *Language codes.*

Description

A data.frame (7879 rows, 2 columns) containing all language codes. The columns are as follows:

Details

- id. id (or code) of the language
- name. longer name of the language

Index