Package ‘ckanr’

October 12, 2022

Title  Client for the Comprehensive Knowledge Archive Network (‘CKAN’) API

Description  Client for ‘CKAN’ API (<https://ckan.org/>). Includes interface to ‘CKAN’ APIs for search, list, show for packages, organizations, and resources. In addition, provides an interface to the ‘datastore’ API.

Version 0.6.0

License MIT + file LICENSE

LazyData true

URL https://docs.ropensci.org/ckanr/ (website)
https://github.com/ropensci/ckanr (devel)

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

Encoding UTF-8

Language en-US

Depends DBI (>= 0.3.1)

Imports methods, stats, utils, crul, jsonlite (>= 0.9.17), dplyr (>= 0.7.0), dbplyr, magrittr

Suggests sf, readxl, testthat, xml2, lazyeval

RoxygenNote 7.1.1

X-schema.org-keywords database, open-data, ckan, api, data, dataset

X-schema.org-applicationCategory Data Access

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

NeedsCompilation no

Author Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>), Imanuel Costigan [aut], Wush Wu [aut] (<https://orcid.org/0000-0001-5180-0567>), Florian Mayer [aut] (<https://orcid.org/0000-0003-4269-4242>), Sharla Gelfand [aut]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2021-02-03 19:30:02 UTC
R topics documented:

ckanr-package .................................................. 3
as.ckan_group .................................................. 5
as.ckan_organization .......................................... 5
as.ckan_package ................................................ 6
as.ckan_related ............................................... 7
as.ckan_resource ............................................... 8
as.ckan_tag ...................................................... 9
as.ckan_user .................................................... 9
changes .......................................................... 10
ckanr-deprecated .............................................. 11
ckanr_settings ................................................ 11
ckan_setup ...................................................... 12
ckan_classes ................................................... 14
ckan_fetch ....................................................... 15
ckan_info ........................................................ 17
dashboard_activity_list ...................................... 18
dashboard_count ................................................ 19
ds_create ......................................................... 20
ds_create_dataset .............................................. 21
ds_search ........................................................ 23
ds_search_sql ................................................... 24
group_create .................................................... 25
group_delete ..................................................... 27
group_list ........................................................ 28
group_patch ...................................................... 29
group_show ........................................................ 30
group_update ..................................................... 31
license_list ...................................................... 32
organization_create ........................................... 33
organization_delete .......................................... 35
organization_list .............................................. 36
organization_purge ............................................ 37
organization_show ............................................ 38
package_activity_list ......................................... 39
package_create ................................................ 40
package_delete ................................................ 42
package_list .................................................... 43
package_list_current ......................................... 44
package_patch .................................................. 45
package_revision_list ......................................... 46
package_search ................................................ 48
package_show .................................................... 50
package_update ................................................ 51
ping .............................................................. 52
related_create ................................................ 53
related_delete ................................................ 54
ckanr-package

R client for the CKAN API

Description

ckanr is a full client for the CKAN API, wrapping all APIs, including for reading and writing data. Please get in touch (https://github.com/ropensci/ckanr/issues or https://discuss.ropensci.org/) if you have problems, or have use cases that we don’t cover yet.

CKAN API

Document for the CKAN API is at https://docs.ckan.org/en/latest/api/index.html. We’ll always be following the lastest version of the API.

ckanr package API

The functions can be grouped into those for setup, packages, resources, tags, organizations, groups, and users.

- Setup - The main one is ckanr_setup() - and many related functions, e.g., get_default_key()
• Packages - Create a package with package_create(), and see other functions starting with package_*
• Resources - Create a package with resource_create(), and see other functions starting with resource_*
• Tags - List tags with tag_list(), and see other functions starting with tag_*
• Organizations - List organizations with organization_list(), show a specific organization with organization_show(), and create with organization_create()
• Groups - List groups with group_list(), and see other functions starting with group_*
• Users - List users with user_list(), and see other functions starting with user_*
• Related items - See functions starting with related_*

Datastore

We are also working on supporting the Datastore extension (https://docs.ckan.org/en/latest/maintaining/datastore.html). We currently have these functions:

• ds_create()
• ds_create_dataset()
• ds_search()
• ds_search_sql()

Fetch

Data can come back in a huge variety of formats. We’ve attempted a function to help you fetch not just metadata but the actual data for a link to a file on a CKAN instance. Though if you know what you’re doing, you can easily use whatever is your preferred tool for the job (e.g., maybe you like read.csv() for reading csv files).

CKAN Instances

We have a helper function (servers()) that spits out the current CKAN instances we know about, with URLs to their base URLs that should work using this package. That is, not necessarily landing pages of each instance, although, the URL may be the landing page and the base API URL.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>
Florian Mayer <florian.wendelin.mayer@gmail.com>
Wush Wu
Imanuel Costigan <i.costigan@me.com>
as.ckan_group

ckan_group class helpers

Usage

as.ckan_group(x, ...)

is.ckan_group(x)

Arguments

x

Variety of things, character, list, or ckan_group class object

...

Further args passed on to group_show() if character given

Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

(grps <- group_list())
grps[[3]]

# create item class from only an item ID
as.ckan_group(grps[[3]]$id)

# gives back itself
(x <- as.ckan_group(grps[[3]]$id))
as.ckan_group(x)

## End(Not run)

as.ckan_organization

ckan_organization class helpers

Usage

as.ckan_organization(x, ...)

is.ckan_organization(x)
Arguments

x  Variety of things, character, list, or ckan_organization class object
...

Further args passed on to organization_show() if character given

Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
key = getOption("ckan_demo_key"))

(orgs <- organization_list(limit = 3))
orgs[[3]]

# create item class from only an item ID
as.ckan_organization(orgs[[3]]$id)

# gives back itself
(x <- as.ckan_organization(orgs[[3]]$id))
as.ckan_organization(x)

## End(Not run)
Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
key = getOption("ckan_demo_key"))

(pkgs <- package_search())
pkgs$results
pkgs$results[[3]]

# create item class from only an item ID
as.ckan_package(pkgs$results[[3]]$id)

# gives back itself
(x <- as.ckan_package(pkgs$results[[3]]$id))
as.ckan_package(x)

## End(Not run)

### as.ckan_related

ckan_related class helpers

Description

ckan_related class helpers

Usage

as.ckan_related(x, ...)

is.ckan_related(x)

Arguments

x Variety of things, character, list, or ckan_related class object
...

Further args passed on to related_show() if character given

Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
key = getOption("ckan_demo_key"))

(x <- package_create("foobbbbarrrrrr") %>%
related_create(title = "my resource",
   type = "visualization"))

# create item class from only an item ID
as.ckan_related(x$id)
as.ckan_resource
can_resource class helpers

Description
ckan_resource class helpers

Usage
as.ckan_resource(x, ...)

is.ckan_resource(x)

Arguments
x Variety of things, character, list, or ckan_package class object
...
Further args passed on to resource_show() if character given

Examples
## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
key = getOption("ckan_demo_key"))

(resrcs <- resource_search(q = 'name:dataset'))
resrcs$results
resrcs$results[[3]]

# create item class from only an item ID
as.ckan_resource(resrcs$results[[3]]$id)

# gives back itself
(x <- as.ckan_resource(resrcs$results[[3]]$id))
as.ckan_resource(x)

## End(Not run)
as.ckan_tag

**Description**

ckan_tag class helpers

**Usage**

as.ckan_tag(x, ...)

is.ckan_tag(x)

**Arguments**

- **x**
  - Variety of things, character, list, or ckan_tag class object
- **...**
  - Further args passed on to `tag_show()` if character given

**Examples**

```r
## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
key = getOption("ckan_demo_key"))

(tags <- tag_search(query = 'ta'))
tags[[3]]

# create item class from only an item ID
as.ckan_tag(tags[[3]]$id)

# gives back itself
(x <- as.ckan_tag(tags[[3]]$id))
as.ckan_tag(x)

## End(Not run)
```

---

as.ckan_user

**Description**

ckan_user class helpers

**Usage**

as.ckan_user(x, ...)

is.ckan_user(x)
changes

Get an activity stream of recently changed datasets on a site.

Description

Get an activity stream of recently changed datasets on a site.

Usage

```
changes(
  offset = 0,
  limit = 31,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

- `offset` (numeric) Where to start getting activity items from (optional, default: 0)
- `limit` (numeric) The maximum number of activities to return (optional, default: 31)
- `url` Base url to use. Default: https://data.ontario.ca/ See also `ckanr_setup` and `get_default_url`.

Examples

```r
## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
  key = getOption("ckan_demo_key"))

(usr <- user_list())
usr[1:3]
usr[3]

# create item class from only an item ID
as.ckan_user(usr[3]$id)

# gives back itself
(x <- as.ckan_user(usr[3]$id))
as.ckan_user(x)

## End(Not run)
```
key: A privileged CKAN API key. Default: your key set with `ckanr_setup`.

as: (character) One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)

... Curl args passed on to verb-POST (optional)

Examples

```r
## Not run:
changes()
changes(as = 'json')
changes(as = 'table')
## End(Not run)
```

---

**ckanr-deprecated**

**Deprecated functions in ckanr**

**Description**

These functions still work but will be removed (defunct) in the next version.

**Details**

- `ds_create_dataset()`: The functionality of this function is already in another function in this package. See function `resource_create()`

---

**ckanr_settings**

**Get or set ckanr CKAN settings**

**Description**

Get or set ckanr CKAN settings

**Usage**

```r
ckanr_settings()
get_default_url()
get_default_key()
get_test_url()
```
get_test_key()
get_test_did()
get_test_rid()
get_test_gid()
get_test_oid()
get_test_behaviour()

Value

ckanr_settings prints your base url, API key (if used), and optional test server settings (URL, API key, a dataset ID and a resource ID). ckanr_setup sets your production and test settings, while get_test_* get each of those respective settings. test Behaviour indicates whether the CKANR test suite will skip ("SKIP") or fail ("FAIL") writing tests in case the configured test CKAN settings don’t work.

See Also

ckanr_setup(), get_default_url(), get_default_key(), get_test_url(), get_test_key(), get_test_did(), get_test_rid(), get_test_gid(), get_test_oid, get_test_behaviour

Examples

ckanr_settings()
ckanr_setup

proxy = NULL
)

Arguments

url A CKAN URL (optional), default: https://data.ontario.ca
key A CKAN API key (optional, character)
test_url (optional, character) A valid CKAN URL for testing purposes
test_key (optional, character) A valid CKAN API key privileged to create datasets at
    test_url
test_did (optional, character) A valid CKAN dataset ID, existing at test_url
test_rid (optional, character) A valid CKAN resource ID, attached to did
test_gid (optional, character) A valid CKAN group name at test_url
test_oid (optional, character) A valid CKAN organization name at test_url
test_behaviour (optional, character) Whether to fail ("FAIL") or skip ("SKIP") writing tests in
    case of problems with the configured test CKAN.
proxy an object of class request from a call to curl::proxy()

Details

ckanr_setup() sets CKAN connection details. ckanr’s functions default to use the default URL
and API key unless specified explicitly.

ckanr’s automated tests require a valid CKAN URL, a privileged API key for that URL, plus the
IDs of an existing dataset and an existing resource, respectively.

The writing tests (create, update, delete) can fail for two reasons: failures in ckanr’s code which the
tests aim to detect, or failures in the configured CKAN, which are not necessarily a problem with
ckanr’s code but prevent the tests to prove otherwise.

Setting test_behaviour to "SKIP" will allow writing tests to skip if the configured test CKAN
fails. This is desirable to e.g. test the other functions even if the tester has no write access to a
CKAN instance.

Setting test_behaviour to "FAIL" will let the tester find any problems with both the configured
test CKAN and the writing functions.

Examples

# CKAN users without admin/editor privileges could run:
ckanr_setup(url = "https://data.ontario.ca/")

# Privileged CKAN editor/admin users can run:
ckanr_setup(url = "https://data.ontario.ca/", key = "some-CKAN-API-key")

# ckanR developers/testers can run:
ckanr_setup(url = "https://data.ontario.ca/", key = "some-CKAN-API-key",
    test_url = "http://test-ckan.gov/", test_key = "test-ckan-API-key",
    test_did = "test-ckan-dataset-id", test_rid = "test-ckan-resource-id",
    test_gid = "test-group-name", test_oid = "test-organization-name",
    test_behaviour = "FAIL")
test Behaviour = "FAIL"

# Not specifying the default CKAN URL will reset the CKAN URL to its default
# "https://data.ontario.ca/"
ckanr_set Up()  

# set a proxy
ckanr_setup(proxy = curl::proxy("64.251.21.73:8080"))
ckanr_settings()  
## run without setting proxy to reset to no proxy
ckanr_setup()
ckanr_settings()

---

### Description

ckan S3 classes

### The classes

- `ckan_package` - CKAN package
- `ckan_resource` - CKAN resource
- `ckan_related` - CKAN related item

### Coercion

The functions `as.ckan_*()` for each CKAN object type coerce something to a S3 class of that type. For example, you can coerce a package ID as a character string into an `ckan_package` object by calling `as.ckan_package(<id>).`

### Testing for classes

To test whether an object is of a particular `ckan_*` class, there is an `is.ckan_*()` function for all of the classes listed above. You can use one of those functions to get a logical back, `TRUE` or `FALSE`.

### Manipulation

These are simple S3 classes, basically an R list with an attached class so we can know what to do with the object and have flexible inputs and outputs from functions. You can edit one of these classes yourself by simply changing values in the list.
**ckan_fetch**  
*Download a file*

**Description**

Download a file

**Usage**

\[
\text{ckan_fetch}(x, \text{store} = \"session\", \text{path} = \"file\", \text{format} = \text{NULL}, \text{key} = \text{get_default_key}(), \ldots)
\]

**Arguments**

- **x** URL for the file
- **store** One of session (default) or disk. session stores in R session, and disk saves the file to disk.
- **path** if store="disk", you must give a path to store file to
- **format** Format of the file. Required if format is not detectable through file URL.
- **key** A CKAN API key (optional, character)
- **...** Arguments passed on to the function used to read the file, if store="session". See details.

**Details**

The ... argument can be used to pass additional arguments to the function that actually reads the file. It is only used when store="session" and if the file is not a ZIP file.

This list shows what function is used to read what file format, so that you can see what additional arguments are available:

- **csv**: read.csv
- **xls, xlsx**: read_excel
- **xml**: read_xml
- **html**: read_html
- **json**: fromJSON
- **shp, geojson**: st_read
- **txt**: read.table
Examples

```r
## Not run:
# CSV file
ckanr_setup("http://datamx.io")
res <- resource_show(id = "6145a539-cbde-4b0d-a3d3-d1a5eb013f5c", as = "table")
head(ckan_fetch(res$url))
ckan_fetch(res$url, "disk", "myfile.csv")

# CSV file, format not available
ckanr_setup("https://ckan0.cf.opendata.inter.prod-toronto.ca")
res <- resource_show(id = "c57c3e1c-20e2-470f-bc82-e39a0264be31", as = "table")
res$url
res$format
head(ckan_fetch(res$url, format = res$format))

# Excel file - requires readxl package
ckanr_setup("http://datamx.io")
res <- resource_show(id = "e883510e-a082-435c-872a-c5b915857ae1", as = "table")
head(ckan_fetch(res$url))

# Excel file, multiple sheets - requires readxl package
ckanr_setup()
res <- resource_show(id = "ce02a1cf-35f1-41df-91d9-11ed1fdd4186", as = "table")
x <- ckan_fetch(res$url)
names(x)
head(x[['Mayor - Maire']])

# XML file - requires xml2 package
# ckanr_setup("http://data.ottawa.ca")
# res <- resource_show(id = "380061c1-6c46-4da6-a01b-7ab0f49a881e", as = "table")
# ckan_fetch(res$url)

# HTML file - requires xml2 package
ckanr_setup("http://open.canada.ca/data/en")
res <- resource_show(id = "80321bac-4283-487c-93bd-c65acaa660f5", as = "table")
ckan_fetch(res$url)
library("xml2")
xml_text(xml_find_first(xml_children(ckan_fetch(res$url))[[1]], "title"))

# JSON file, by default reads in to a data.frame for ease of use
ckanr_setup("http://data.surrey.ca")
res <- resource_show(id = "8d07c662-800d-4977-9e3e-5a3d2d1e99ab", as = "table")
head(ckan_fetch(res$url))

# SHP file (spatial data, ESRI format) - requires sf package
```
ckanr_setup("https://ckan0.cf.opendata.inter.prod-toronto.ca")
res <- resource_show(id = "27362290-8bbf-434b-a9de-325a6c2ef923", as = "table")
x <- ckan_fetch(res$url)
class(x)
plot(x[, c("AREA_NAME", "geometry")])

# GeoJSON file - requires sf package
ckanr_setup("http://datamx.io")
res <- resource_show(id = "b1cd35b7-479e-4fa0-86e9-e897d3c617e6", as = "table")
x <- ckan_fetch(res$url)
class(x)
plot(x[, c("mun_name", "geometry")])

# ZIP file - packages required depends on contents
ckanr_setup("https://ckan0.cf.opendata.inter.prod-toronto.ca")
res <- resource_show(id = "bb21e1b8-a466-41c6-8bc3-3c362cb1ed55", as = "table")
x <- ckan_fetch(res$url)
names(x)
head(x["ChickenpoxAgegroups2017.csv"])]

# TXT file
ckanr_setup("https://ckan0.cf.opendata.inter.prod-toronto.ca")
res <- resource_show(id = "e4211f49-611f-438c-a444-aaa7f3f84117", as = "table")
x <- ckan_fetch(res$url)
head(x)

# TXT file, semicolon used as separator
ckanr_setup("https://data.coat.no")
res <- resource_show(id = "384fe537-e0bd-4e57-8a0d-420b7a745196", as = "table")
x <- ckan_fetch(res$url, sep = ";")
head(x)

## End(Not run)

---

**ckan_info**

*Get information on a CKAN server*

**Description**

Get information on a CKAN server

**Usage**

```r
ckan_info(url = get_default_url(), ...)
ckan_version(url = get_default_url(), ...)
```
Arguments

url Base url to use. Default: https://data.ontario.ca. See also ckanr_setup() and get_default_url(). (required)

... Curl args passed on to crud::verb-GET (optional)

Value

for ckan_info a list with many slots with various info. for ckan_version, list of length two, with actual version as character, and another with version converted to numeric (any dots or letters removed)

Examples

```r
## Not run:
ckan_info()
ckan_info(servers()[5])
ckan_version(servers()[5])
## End(Not run)
```

dashboard_activity_list

*Authorized user’s dashboard activity stream*

Description

Authorized user’s dashboard activity stream

Usage

```r
dashboard_activity_list(
  limit = 31,
  offset = 0,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

limit (integer) The maximum number of activities to return (optional). Default: 31
offset (integer) Where to start getting activity items from (optional). Default: 0
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
A privileged CKAN API key. Default: your key set with `ckanr_setup`.

One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)

Curl args passed on to `verb-POST` (optional)

### Examples

```r
## Not run:
## Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

## get activity
(res <- dashboard_activity_list())

## End(Not run)
```

### Description

Number of new activities of an authorized user

### Usage

```r
dashboard_count(
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

### Arguments

- **url**: Base url to use. Default: `https://data.ontario.ca` See also `ckanr_setup` and `get_default_url`.
- **key**: A privileged CKAN API key. Default: your key set with `ckanr_setup`.
- **as**: (character) One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
- **...**: Curl args passed on to `verb-POST` (optional)
ds_create

Details

Important: Activities from the user herself are not counted by this function even though they appear in the dashboard (users don’t want to be notified about things they did themselves).

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# count
dashboard_count()

## End(Not run)
```

ds_create  
*Add a new table to a datastore*

Description

BEWARE: This function still doesn’t quite work yet.

Usage

```r
ds_create(
  resource_id = NULL,
  resource = NULL,
  force = FALSE,
  aliases = NULL,
  fields = NULL,
  records = NULL,
  primary_key = NULL,
  indexes = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource_id</td>
<td>(string) Resource id that the data is going to be stored against.</td>
</tr>
<tr>
<td>resource</td>
<td>(dictionary) Resource dictionary that is passed to <code>resource_create()</code>. Use instead of <code>resource_id</code> (optional)</td>
</tr>
<tr>
<td>force</td>
<td>(logical) Set to TRUE to edit a read-only resource. Default: FALSE</td>
</tr>
<tr>
<td>aliases</td>
<td>(character) Names for read only aliases of the resource. (optional)</td>
</tr>
</tbody>
</table>
ds_create_dataset

Description

Datstore - create a new resource on an existing dataset

fields (list) Fields/columns and their extra metadata. (optional)
records (list) The data, eg: [{"dob": "2005", "some_stuff": ["a", "b"]}] (optional)
primary_key (character) Fields that represent a unique key (optional)
indexes (character) Indexes on table (optional)
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb-POST (optional)

References


Examples

```r
## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("foobarrrr", author="Jane Doe"))

# then create a resource
file <- system.file("examples", "actinidiaceae.csv", package = "ckanr")
(xx <- resource_create(package_id = res$id,
description = "my resource",
name = "bears",
upload = file,
rcurl = "http://google.com")
)

ds_create(resource_id = xx$id, records = iris, force = TRUE)
resource_show(xx$id)

## End(Not run)
```
ds_create_dataset

Usage

ds_create_dataset(
    package_id, 
    name, 
    path, 
    url = get_default_url(), 
    key = get_default_key(), 
    as = "list", 
    ...
)

Arguments

package_id  (character) Existing package ID (required)
name        (character) Name of the new resource (required)
path        (character) Path of the file to add (required)
url         Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key         A privileged CKAN API key. Default: your key set with ckanr_setup
as          (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
...         Curl args passed on to verb-POST (optional)

Details

This function is deprecated - will be defunct in the next version of this package

References

http://docs.ckan.org/en/latest/api/index.html#ckan.logic.action.create.resource_create

Examples

## Not run:
path <- system.file("examples", "actinidiaceae.csv", package = "ckan")
ckanr_setup(url = "https://demo.ckan.org/", key = "my-demo-ckan-org-api-key")
ds_create_dataset(package_id='testingagain', name="mydata", path = path)

# Testing: see ?ckanr_setup to set test settings
ckanr_setup(test_url = "http://my-ckan.org/",
             test_key = "my-ckan-api-key",
             test_did="an-existing-package-id",
             test_rid="an-existing-resource-id")
ds_create_dataset(package_id=get_test_pid(), name="mydata",
                   path=system.file("examples",
                                    "actinidiaceae.csv",
                                    ...)
package = "ckanr"),
  key = get_test_key(),
  url = get_test_url())

## End(Not run)

---

### ds_search

**Datastore - search or get a dataset from CKRAN datastore**

#### Description

Datastore - search or get a dataset from CKRAN datastore

#### Usage

```r
ds_search(
  resource_id = NULL,
  filters = NULL,
  q = NULL,
  plain = NULL,
  language = NULL,
  fields = NULL,
  offset = NULL,
  limit = NULL,
  sort = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

#### Arguments

- `resource_id` (character) id or alias of the resource to be searched against
- `filters` (character) matching conditions to select, e.g. `{"key1": "a", "key2": "b"}` (optional)
- `q` (character) full text query (optional)
- `plain` (character) treat as plain text query (optional, default: TRUE)
- `language` (character) language of the full text query (optional, default: english)
- `fields` (character) fields to return (optional, default: all fields in original order)
- `offset` (numeric) Where to start getting activity items from (optional, default: 0)
- `limit` (numeric) The maximum number of activities to return (optional, default: 100)
- `sort` Field to sort on. You can specify ascending (e.g., score desc) or descending (e.g., score asc), sort by two fields (e.g., score desc, price asc), or sort by a function (e.g., sum(x_f, y_f) desc, which sorts by the sum of x_f and y_f in a descending order). (optional)
ds_search_sql

url  Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.

key  A privileged CKAN API key. Default: your key set with ckanr_setup

as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)

...  Curl args passed on to verb=POST (optional)

Details
From the help for this method “The datastore_search action allows you to search data in a resource. DataStore resources that belong to private CKAN resource can only be read by you if you have access to the CKAN resource and send the appropriate authorization.”

Setting plain=FALSE enables the entire PostgreSQL full text search query language. A listing of all available resources can be found at the alias table_metadata full text search query language: http://www.postgresql.org/docs/9.1/static/datatype-textsearch.html#DATATYPE-TSQUERY

Examples
## Not run:
ckanr_setup(url = 'https://data.nhm.ac.uk/')

ds_search(resource_id = '8f0784a6-82dd-44e7-b105-6194e046eb8d')
ds_search(resource_id = '8f0784a6-82dd-44e7-b105-6194e046eb8d',
  as = "table")
ds_search(resource_id = '8f0784a6-82dd-44e7-b105-6194e046eb8d',
  as = "json")
ds_search(resource_id = '8f0784a6-82dd-44e7-b105-6194e046eb8d', limit = 1,
  as = "table")
ds_search(resource_id = '8f0784a6-82dd-44e7-b105-6194e046eb8d', q = "a*")
## End(Not run)

ds_search_sql

Datastore - search or get a dataset from CKRAN datastore

Description
Datastore - search or get a dataset from CKRAN datastore

Usage
ds_search_sql(
  sql,
  url = get_default_url(),
)
group_create

key = get_default_key(),
as = "list",
...)

Arguments

sql (character) A single SQL select statement. (required)
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb-POST (optional)

Examples

## Not run:
url <- 'https://demo.ckan.org/
sql <- 'SELECT * from "f4129802-22aa-4437-b9f9-8a8f3b7b2a53" LIMIT 2'
ds_search_sql(sql, url = url, as = "table")
sql2 <- 'SELECT "Species","Genus","Family" from "f4129802-22aa-4437-b9f9-8a8f3b7b2a53" LIMIT 2'
ds_search_sql(sql2, url = url, as = "table")

## End(Not run)

group_create

Create a group

Description

Create a group

Usage

group_create(
  name = NULL,
  id = NULL,
  title = NULL,
  description = NULL,
  image_url = NULL,
  type = NULL,
  state = "active",
  approval_status = NULL,
  extras = NULL,
packages = NULL,
groups = NULL,
users = NULL,
url = get_default_url(),
key = get_default_key(),
as = "list",
...
)

Arguments

name (character) the name of the new dataset, must be between 2 and 100 characters long and contain only lowercase alphanumeric characters,

• and _, e.g. 'warandpeace'

id (character) The id of the group (optional)
title (character) The title of the dataset (optional, default: same as name)
description (character) The description of the group (optional)
image_url (character) The URL to an image to be displayed on the group’s page (optional)
type (character) The type of the dataset (optional), IDatasetForm plugins associate themselves with different dataset types and provide custom dataset handling behaviour for these types
state (character) The current state of the dataset, e.g. 'active' or 'deleted', only active datasets show up in search results and other lists of datasets, this parameter will be ignored if you are not authorized to change the state of the dataset (optional, default: 'active')
approval_status (character) Approval status (optional)
extras (list of dataset extra dictionaries) The dataset’s extras (optional), extras are arbitrary (key: value) metadata items that can be added to datasets, each extra dictionary should have keys 'key' (a string), 'value' (a string)
packages (data.frame) The datasets (packages) that belong to the group, a data.frame, where each row has column 'name' (string, the id or name of the dataset) and optionally 'title' (string, the title of the dataset)
groups (data.frame) The groups to which the dataset belongs (optional), each data.frame row should have one or more of the following columns which identify an existing group: 'id' (the id of the group, string), or 'name' (the name of the group, string), to see which groups exist call group_list()
users (list of dictionaries) The users that belong to the group, a list of dictionaries each with key 'name' (string, the id or name of the user) and optionally 'capacity' (string, the capacity in which the user is a member of the group)
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup
group_delete

Delete a group

Description
Delete a group

Usage
group_delete(id, url = get_default_url(), key = get_default_key(), ...)

Arguments
- id (character) The id of the group. Required.
- url Base url to use. Default: https://data.ontario.ca See also ckanr_setup() and get_default_url()
- key A privileged CKAN API key. Default: your key set with ckanr_setup
- ... Curl args passed on to crul::verb-POST (optional)

Examples
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org", key = getOption("ckan_demo_key"))

# create a group
(res <- group_create("fruitloops2", description="A group about fruitloops"))
res$users
res$num_followers

## End(Not run)
group_show(res$id)

# delete the group
group_delete(res)
## or with it's id
# group_delete(res$id)

## End(Not run)

---

group_list

List groups.

Description

List groups.

Usage

```r
group_list(
  offset = 0,
  limit = 31,
  sort = NULL,
  groups = NULL,
  all_fields = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

- `offset` (numeric) Where to start getting activity items from (optional, default: 0)
- `limit` (numeric) The maximum number of activities to return (optional, default: 31)
- `sort` Field to sort on. You can specify ascending (e.g., score desc) or descending (e.g., score asc), sort by two fields (e.g., score desc, price asc), or sort by a function (e.g., `sum(x_f, y_f) desc`, which sorts by the sum of `x_f` and `y_f` in a descending order).
- `groups` (character) A list of names of the groups to return, if given only groups whose names are in this list will be returned
- `all_fields` (logical) Return full group dictionaries instead of just names. Default: FALSE
- `url` Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- `key` A privileged CKAN API key, Default: your key set with ckanr_setup
group_patch

as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)

... Curl args passed on to verb=POST (optional)

Examples

## Not run:
group_list(limit = 3)
group_list(limit = 3, as = 'json')
group_list(limit = 3, as = 'table')

## End(Not run)

---

**group_patch**

*Update a group's metadata*

Description

Update a group's metadata

Usage

```r
group_patch(
x,
id,
url = get_default_url(),
key = get_default_key(),
as = "list",
...
)
```

Arguments

- `x` (list) A list with key-value pairs
- `id` (character) Resource ID to update (required)
- `url` Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- `key` A privileged CKAN API key. Default: your key set with ckanr_setup
- `as` (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
- `...` Curl args passed on to verb=POST (optional)
Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org", key = getOption("ckan_demo_key"))

# Create a package
(res <- group_create("hello-my-world2"))

# Get a resource
grp <- group_show(res$id)
grp$title
grp$author_email

# Make some changes
x <- list(title = "!hello world!", maintainer_email = "hello@world.com")
group_patch(x, id = grp)

## End(Not run)
```

---

**group_show**

*Show a package*

**Description**

Show a package

**Usage**

```r
group_show(
  id,
  include_datasets = TRUE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

**Arguments**

- `id` (character) Package identifier.
- `include_datasets` (logical) Include a list of the group’s datasets. Default: `TRUE`
- `url` Base url to use. Default: `https://data.ontario.ca/` See also `ckanr_setup` and `get_default_url`.
- `key` A privileged CKAN API key. Default: your key set with `ckanr_setup`
### group_update

**Update a group**

**Description**

Update a group

**Usage**

```r

group_update(
  x,
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

**Details**

By default the help and success slots are dropped, and only the result slot is returned. You can request raw json with `as = 'json'` then parse yourself to get the help slot.

**Examples**

```r

## Not run:
res <- group_list()

# via a group name/id
group_show(res[[1]]$name)

# or via an object of class ckan_group
group_show(res[[1]])

# return different data formats

# or via an object of class ckan_group
group_show(res[[1]]$name, as = 'json')

# or via an object of class ckan_group

## End(Not run)
```

...
license_list

Return the list of licenses available for datasets on the site.

Description

Return the list of licenses available for datasets on the site.

Usage

license_list(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
organization_create

Arguments

id  (character) Package identifier.
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb-POST (optional)

Examples

## Not run:
license_list()
license_list(as = "table")
license_list(as = "json")

## End(Not run)

organization_create Create an organization

Description

Create an organization

Usage

organization_create(  
  name = NULL,
  id = NULL,
  title = NULL,
  description = NULL,
  image_url = NULL,
  state = "active",
  approval_status = NULL,
  extras = NULL,
  packages = NULL,
  users = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)


organization_create

Arguments

- **name** (character) the name of the organization, a string between 2 and 100 characters long, containing only lowercase alphanumeric characters, and _
- **id** (optional) the id of the organization
- **title** (character) the title of the organization (optional)
- **description** (character) the description of the organization (optional)
- **image_url** (character) the URL to an image to be displayed on the organization’s page (optional)
- **state** (character) the current state of the organization, e.g. 'active' or 'deleted', only active organization show up in search results and other lists of organization, this parameter will be ignored if you are not authorized to change the state of the organization (optional). Default: 'active'
- **approval_status** (character) Approval status
- **extras** The organization’s extras (optional), extras are arbitrary (key: value) metadata items that can be added to organizations, each extra dictionary should have keys 'key' (a string), 'value' (a string) package_relationship_create for the format of relationship dictionaries (optional)
- **packages** (list of dictionaries) the datasets (packages) that belong to the organization, a list of dictionaries each with keys 'name' (string, the id or name of the dataset) and optionally 'title' (string, the title of the dataset)
- **users** (character) the users that belong to the organization, a list of dictionaries each with key 'name' (string, the id or name of the user) and optionally 'capacity' (string, the capacity in which the user is a member of the organization)
- **url** Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- **key** A privileged CKAN API key, Default: your key set with ckanr_setup as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
- **...** Curl args passed on to verb=POST (optional)

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create an organization
(res <- organization_create("foobar", title = "Foo bars",
                           description = "love foo bars"))
res$name

## End(Not run)
```
**organization_delete**  
*Delete an organization*

**Description**
Delete an organization

**Usage**
```
organization_delete(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

**Arguments**
- **id**  
  (character) name or id of the organization
- **url**  
  Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- **key**  
  A privileged CKAN API key, Default: your key set with ckanr_setup
- **as**  
  (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
- **...**  
  Curl args passed on to verb-POST (optional)

**Value**
an empty list on success

**Examples**
```
## Not run:
ckanr_setup(url = "https://demo.ckan.org", key=getOption("ckan_demo_key"))

# create an organization
(res <- organization_create("foobar", title = "Foo bars", 
  description = "love foo bars"))

# delete the organization just created
res$id
organization_delete(id = res$id)

## End(Not run)
```
organization_list  List organization

Description
List organization

Usage

organization_list(
  order_by = c("name", "package"),
  decreasing = FALSE,
  organizations = NULL,
  all_fields = TRUE,
  limit = 31,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

order_by  (character, only the first element is used). The field to sort the list by, must be name or packages.
decreasing  (logical). Is the sort-order is decreasing or not.
organizations  (character or NULL). A list of names of the organizations to return. NULL returns all organizations.
all_fields  (logical). Return the name or all fields of the object.
limit  (numeric) The maximum number of organizations to return (optional, default: 31)
url  Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key  A privileged CKAN API key, Default: your key set with ckanr_setup
as  (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
...  Curl args passed on to verb-POST (optional)

Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org/")
organization_purge

Purge an organization

Description

IMPORTANT: You must be a sysadmin to purge an organization. Once an organization is purged, it is permanently removed from the system.

Usage

organization_purge(id, url = get_default_url(), key = get_default_key(), as = "list", ...)

Arguments

id (character) name or id of the organization
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb=POST (optional)

Value

an empty list on success
organization_show

Examples

```r
# Not run:
ckanr_setup(url = "https://demo.ckan.org", key=getOption("ckan_demo_key"))

# create an organization
(res <- organization_create("foobar", title = "Foo bars",
                               description = "love foo bars"))

# delete the organization just created
res$id
organization_delete(id = res$id)

# purge the organization just deleted
res$id
organization_purge(id = res$id)

# End(Not run)
```

organization_show Show an organization

Description

Show an organization

Usage

```r
organization_show(
  id,
  include_datasets = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

- `id` (character) Organization id or name.
- `include_datasets` (logical). Whether to include a list of the organization datasets
- `url` Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- `key` A privileged CKAN API key. Default: your key set with ckanr_setup
- `as` (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
- `...` Curl args passed on to verb=POST (optional)
package_activity_list

Details
By default the help and success slots are dropped, and only the result slot is returned. You can request raw json with `as = 'json'` then parse yourself to get the help slot.

Examples

```r
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

res <- organization_create("stuffthings2")
organization_show(res$id)

## End(Not run)
```

Description

Return a list of the package’s activity

Usage

```r
package_activity_list(
  id,
  offset = 0,
  limit = 31,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...)
```

Arguments

- `id` (character) Package identifier.
- `offset` (numeric) Where to start getting activity items from (optional, default: 0)
- `limit` (numeric) The maximum number of activities to return (optional, default: 31)
- `url` Base url to use. Default: https://data.ontario.ca/ See also `ckanr_setup` and `get_default_url`.
- `key` A privileged CKAN API key. Default: your key set with `ckanr_setup`
- `as` (character) One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
- `...` Curl args passed on to `verb=POST` (optional)
Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("owls64"))

# list package activity
package_activity_list(res$id)

# make a change
x <- list(maintainer = "Jane Forest")
package_update(x, res)

# list activity again
package_activity_list(res)

# output different data formats
package_activity_list(res$id, as = "table")
package_activity_list(res$id, as = "json")

## End(Not run)
```

---

**package_create**

Create a package

### Description

Create a package

### Usage

```r
package_create(
  name = NULL,
  title = NULL,
  private = FALSE,
  author = NULL,
  author_email = NULL,
  maintainer = NULL,
  maintainer_email = NULL,
  license_id = NULL,
  notes = NULL,
  package_url = NULL,
  version = NULL,
  state = "active",
  type = NULL,
  resources = NULL,
```

package_create

tags = NULL,
extras = NULL,
relationships_as_object = NULL,
relationships_as_subject = NULL,
groups = NULL,
owner_org = NULL,
url = get_default_url(),
key = get_default_key(),
as = "list",
)

Arguments

name (character) the name of the new dataset, must be between 2 and 100 characters long and contain only lowercase alphanumeric characters, - and _, e.g. 'warand-peace'
title (character) the title of the dataset (optional, default: same as name)
private (logical) whether the dataset should be private (optional, default: FALSE), requires a value for owner_org if TRUE
author (character) the name of the dataset’s author (optional)
author_email (character) the email address of the dataset’s author (optional)
maintainer (character) the name of the dataset’s maintainer (optional)
maintainer_email (character) the email address of the dataset’s maintainer (optional)
license_id (license id string) - the id of the dataset’s license, see license_list() for available values (optional)
notes (character) a description of the dataset (optional)
package_url (character) a URL for the dataset’s source (optional)
version (string, no longer than 100 characters) - (optional)
state (character) the current state of the dataset, e.g. 'active' or 'deleted', only active datasets show up in search results and other lists of datasets, this parameter will be ignored if you are not authorized to change the state of the dataset (optional, default: 'active')
type (character) the type of the dataset (optional), IDatasetForm plugins associate themselves with different dataset types and provide custom dataset handling behaviour for these types
resources (list of resource dictionaries) - the dataset’s resources, see resource_create() for the format of resource dictionaries (optional)
tags (list of tag dictionaries) - the dataset’s tags, see tag_create() for the format of tag dictionaries (optional)
extras (list of dataset extra dictionaries) - the dataset’s extras (optional), extras are arbitrary (key: value) metadata items that can be added to datasets, each extra dictionary should have keys 'key' (a string), 'value' (a string)
Delete a package

**Description**

Delete a package
**Usage**

```
package_delete(id, url = get_default_url(), key = get_default_key(), ...)
```

**Arguments**

- **id** (character) The id of the package. Required.
- **url** Base url to use. Default: https://data.ontario.ca See also `ckanr_setup()` and `get_default_url()`
- **key** A privileged CKAN API key. Default: your key set with `ckanr_setup`
- **...** Curl args passed on to `crl::verb-POST` (optional)

**Examples**

```
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org", key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("lions-bears-tigers"))

# show the package
package_show(res)

# delete the package
package_delete(res)

## End(Not run)
```

---

**package_list**

`List datasets.`

**Description**

List datasets.

**Usage**

```
package_list(
    offset = 0,
    limit = 31,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)
```
package_list_current

Arguments

offset  (numeric) Where to start getting activity items from (optional, default: 0)
limit   (numeric) The maximum number of activities to return (optional, default: 31)
url     Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key     A privileged CKAN API key, Default: your key set with ckanr_setup
as      (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
...     Curl args passed on to verb=POST (optional)

Examples

## Not run:
package_list()
package_list(as = 'json')
package_list(as = 'table')

package_list(url = 'https://data.nhm.ac.uk')

## End(Not run)

package_list_current  List current packages with resources.

Description

List current packages with resources.

Usage

package_list_current(
  offset = 0,
  limit = 31,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
package_patch

Update a package’s metadata

Description

Update a package’s metadata

Usage

```r
package_patch(
  x,
  id = NULL,
  extras = NULL,
  http_method = "GET",
  key = get_default_key(),
  url = get_default_url(),
  as = "list",
  ...
)
```
Arguments

- **x** (list) A list with key-value pairs
- **id** (character) Resource ID to update (optional, required if x does not have an "id" field)
- **extras** (character vector) - the dataset’s extras (optional), extras are arbitrary (key: value) metadata items that can be added to datasets, each extra dictionary should have keys ‘key’ (a string), ‘value’ (a string)
- **http_method** (character) which HTTP method (verb) to use; one of "GET" or "POST". Default: "GET"
- **key** A privileged CKAN API key. Default: your key set with **ckanr_setup**
- **url** Base url to use. Default: https://data.ontario.ca/ See also **ckanr_setup** and **get_default_url**.
- **as** (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
- **...** Curl args passed on to verb-POST (optional)

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org", key = getOption("ckan_demo_key"))

# Create a package
(res <- package_create("hello-world13", author="Jane Doe"))

# Get a resource
res <- package_show(res$id)
res$title

# patch
package_patch(res, extras = list(list(key = "foo", value = "bar")))
unclass(package_show(res))

## End(Not run)
```

**package_revision_list**

Return a dataset (package’s) revisions as a list of dictionaries.

**Description**

Return a dataset (package’s) revisions as a list of dictionaries.
Usage

package_revision_list(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

id (character) Package identifier.
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and
get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses
jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to
parse data to data.frame's when possible, so the result can vary from a vector,
list or data.frame. (required)
...
Curl args passed on to verb-POST (optional)

Examples

## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("dolphins"))

# list package revisions
package_revision_list(res$id)

# Make change to the package
x <- list(title = "dolphins and things")
package_patch(x, id = res$id)

# list package revisions
package_revision_list(res$id)

# Output different formats
package_revision_list(res$id, as = "table")
package_revision_list(res$id, as = "json")

## End(Not run)
package_search  

Search for packages.

Description

Search for packages.

Usage

package_search(
  q = "*:*",
  fq = NULL,
  sort = NULL,
  rows = NULL,
  start = NULL,
  facet = FALSE,
  facet.limit = NULL,
  facet.field = NULL,
  facet.mincount = NULL,
  include_drafts = FALSE,
  include_private = FALSE,
  use_default_schema = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

q  
Query terms, defaults to ':', or everything.

fq  
Filter query, this does not affect the search, only what gets returned

sort  
Field to sort on. You can specify ascending (e.g., score desc) or descending (e.g., score asc), sort by two fields (e.g., score desc, price asc), or sort by a function (e.g., sum(x_f, y_f) desc, which sorts by the sum of x_f and y_f in a descending order).

rows  
Number of records to return. Defaults to 10.

start  
Record to start at, default to beginning.

facet  
(logical) Whether to return facet results or not. Default: FALSE

facet.limit  
(numeric) This param indicates the maximum number of constraint counts that should be returned for the facet fields. A negative value means unlimited. Default: 100. Can be specified on a per field basis.

facet.field  
(character) This param allows you to specify a field which should be treated as a facet. It will iterate over each Term in the field and generate a facet count using that Term as the constraint. This parameter can be specified multiple times to
package_search

indicate multiple facet fields. None of the other params in this section will have
any effect without specifying at least one field name using this param.

facet.mincount  (integer) the minimum counts for facet fields should be included in the results

include_drafts (logical) if TRUE draft datasets will be included. A user will only be returned
their own draft datasets, and a sysadmin will be returned all draft datasets. default: FALSE first CKAN version: 2.6.1; dropped from request if CKAN version is older or if CKAN version isn’t available via ckan_version()

include_private (logical) if TRUE private datasets will be included. Only private datasets from the
user’s organizations will be returned and sysadmins will be returned all private
datasets. default: FALSE first CKAN version: 2.6.1; dropped from request if CKAN version is older or if CKAN version isn’t available via ckan_version()

use_default_schema (logical) use default package schema instead of a custom schema defined with
an IDatasetForm plugin. default: FALSE first CKAN version: 2.3.5; dropped from request if CKAN version is older or if CKAN version isn’t available via ckan_version()

url  Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and
get_default_url.

key  A privileged CKAN API key. Default: your key set with ckanr_setup

as (character) One of list (default), table, or json. Parsing with table option uses
jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to
parse data to data.frame’s when possible, so the result can vary from a vector,
list or data.frame. (required)

...  Curl args passed on to verb-POST (optional)

Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org", key=getOption("ckan_demo_key"))

package_search(q = '*:*)
package_search(q = '*:x', rows = 2, as = 'json')
package_search(q = 'x:*', rows = 2, as = 'table')

package_search(q = 'x:*', sort = 'score asc')
package_search(q = 'x:*', fq = 'num_tags:[3 TO x]')$count
package_search(q = 'x:*', fq = 'num_tags:[2 TO x]')$count
package_search(q = 'x:*', fq = 'num_tags:[1 TO x]')$count

## End(Not run)
package_show

Show a package.

Description

Show a package.

Usage

package_show(
id,
use_default_schema = FALSE,
http_method = "GET",
url = get_default_url(),
key = get_default_key(),
as = "list",
...
)

Arguments

id (character/ckan_package) Package identifier, or a ckan_package object
use_default_schema (logical) Use default package schema instead of a custom schema defined with an IDatasetForm plugin. Default: FALSE
http_method (character) which HTTP method (verb) to use; one of "GET" or "POST". Default: "GET"
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb=POST (optional)

Details

By default the help and success slots are dropped, and only the result slot is returned. You can request raw json with as = 'json' then parse yourself to get the help slot.
## Not run:
### Setup
ckanr_setup(url = "https://demo.ckan.org/",
    key = getOption("ckan_demo_key"))

### create a package
(res <- package_create("purposeful55"))

### show package
### From the output of package_create
package_show(res)
### Or, from the ID
package_show(res$id)

### get data back in different formats
package_show(res$id, as = 'json')
package_show(res$id, as = 'table')

### use default schema or not
package_show(res$id, TRUE)

### End(Not run)

---

package_update

**Update a package**

---

### Description

Update a package

### Usage

package_update(
    x,
    id,
    http_method = "GET",
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

### Arguments

- **x** (list) A list with key-value pairs
- **id** (character) Package identifier
http_method (character) which HTTP method (verb) to use; one of "GET" or "POST". Default: "GET"

url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.

key A privileged CKAN API key. Default: your key set with ckanr_setup

as (character) One of list (default), table, or json. Parsing with table option uses
jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to
parse data to data.frame's when possible, so the result can vary from a vector,
list or data.frame. (required)

... Curl args passed on to verb=POST (optional)

Examples

## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# Create a package
(pkg <- package_create("hello-world11", author="Jane Doe"))

# Next show the package to see the fields
(res <- package_show(pkg$id))

## update just chosen things
# Make some changes
x <- list(maintainer_email = "heythere2@things.com")

# Then update the packge
package_update(x, pkg$id)

## End(Not run)

---

ping Ping a CKAN server to test that it’s up or down.

Description

Ping a CKAN server to test that it’s up or down.

Usage

ping(url = get_default_url(), key = get_default_key(), as = "logical", ...)


related_create

Arguments

url
Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.

key
A privileged CKAN API key, Default: your key set with ckanr_setup

as
(character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)

... Curl args passed on to verb=POST (optional)

Examples

## Not run:
ping()
ping(as = "json")

## End(Not run)

related_create

Create a related item

Description

Create a related item

Usage

related_create(
  id,
  title,
  type,
  description = NULL,
  related_id = NULL,
  related_url = NULL,
  image_url = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

id
(character) id of package that the related item should be added to. This should be an alphanumeric string. Required.

title
(character) Title of the related item. Required.
related_delete

Delete a related item.

**Description**

Delete a related item.

**Usage**

related_delete(id, url = get_default_url(), key = get_default_key(), ...)

- **type** (character) The type of the related item. One of API, application, idea, news article, paper, post or visualization. Required.
- **description** (character) description (optional). Optional
- **related_id** (character) An id to assign to the related item. If blank, an ID will be assigned for you. Optional
- **related_url** (character) A url to associated with the related item. Optional
- **image_url** (character) A url to associated image. Optional
- **url** Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- **key** A privileged CKAN API key. Default: your key set with ckanr_setup
- **as** (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("hello-mars"))

# create a related item
related_create(res, title = "asdfdaf", type = "idea")

# pipe operations together
package_create("foobbbbbarrrrr") %>%
  related_create(title = "my resource",
               type = "visualization")

## End(Not run)
```
Arguments

id (character) Resource identifier.
url Base url to use. Default: https://data.ontario.ca See also ckanr_setup() and get_default_url().
key A privileged CKAN API key. Default: your key set with ckanr_setup
... Curl args passed on to curl::verb-POST (optional)

Examples

## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package and a related item
res <- package_create("hello-venus2") %>%
  related_create(title = "my resource",
                 type = "visualization")

# show the related item
related_delete(res)
## or with id itself:
## related_delete(res$id)

## End(Not run)

---

related_list List related items

Description

List related items

Usage

related_list(
  offset = 0,
  limit = 31,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

**related_show**

**Description**

Show a related item

**Usage**

```r
related_show(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

**Arguments**

- `id` (character) Related item identifier.
- `url` Base url to use. Default: `https://data.ontario.ca/` See also `ckanr_setup` and `get_default_url`.
- `key` A privileged CKAN API key. Default: your key set with `ckanr_setup`.
**related_update**

Update a related item

---

**Description**

Update a related item

**Usage**

```r
related_update(
  id,
  title,
  type,
  description = NULL,
  related_id = NULL,
  related_url = NULL,
  image_url = NULL,
)```

---

**as**

(character) One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)

...Curl args passed on to `verb=POST` (optional)

**Details**

By default the help and success slots are dropped, and only the result slot is returned. You can request raw json with `as = 'json'` then parse yourself to get the help slot.

**Examples**

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package and a related item
res <- package_create("hello-pluto2") %>%
  related_create(title = "my resource",
                 type = "visualization")

# show the related item
related_show(res)
related_show(res$id)

# get data back in different formats
related_show(res, as = 'json')
related_show(res, as = 'table')

## End(Not run)
```
```r
url = get_default_url(),
key = get_default_key(),
as = "list",
...
)

Arguments

id (character) id of related item to update. This should be an alphanumeric string. Required.
title (character) Title of the related item. Required.
type (character) The type of the related item. One of API, application, idea, news article, paper, post or visualization. Required.
description (character) description (optional). Optional
related_id (character) An id to assign to the related item. If blank, an ID will be assigned for you. Optional
related_url (character) A url to associated with the related item. Optional
image_url (character) A url to associated image. Optional
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
...

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package and related item
res <- package_create("hello-saturn2") %>%
  related_create(title = "my resource",
                 type = "visualization")

# update the related item
related_update(res, title = "her resource", type = "idea")
```

## End(Not run)
resource_create

Create a resource

Description
Create a resource

Usage

resource_create(
    package_id = NULL,
    rcurl = NULL,
    revision_id = NULL,
    description = NULL,
    format = NULL,
    hash = NULL,
    name = NULL,
    resource_type = NULL,
    mimetype = NULL,
    mimetype_inner = NULL,
    webstore_url = NULL,
    cache_url = NULL,
    size = NULL,
    created = NULL,
    last_modified = NULL,
    cache_last_updated = NULL,
    webstore_last_updated = NULL,
    upload = NULL,
    extras = NULL,
    http_method = "GET",
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>package_id</td>
<td>(character) id of package that the resource should be added to. This should be an alphanumeric string. Required.</td>
</tr>
<tr>
<td>rcurl</td>
<td>(character) url of resource. Required.</td>
</tr>
<tr>
<td>revision_id</td>
<td>(character) revision id (optional)</td>
</tr>
<tr>
<td>description</td>
<td>(character) description (optional). Required.</td>
</tr>
<tr>
<td>format</td>
<td>(character) format (optional)</td>
</tr>
<tr>
<td>hash</td>
<td>(character) hash (optional)</td>
</tr>
</tbody>
</table>
name  (character) name (optional). Required.
resource_type  (character) resource type (optional)
mimetype  (character) mime type (optional)
mimetype_inner  (character) mime type inner (optional)
webstore_url  (character) webstore url (optional)
cache_url  (character) cache url (optional)
size  (integer) size (optional)
created  (character) iso date string (optional)
last_modified  (character) iso date string (optional)
cache_last_updated  (character) iso date string (optional)
webstore_last_updated  (character) iso date string (optional)
upload  (character) A path to a local file (optional)
extras  (list) - the resources' extra metadata fields (optional)
http_method  (character) which HTTP method (verb) to use; one of "GET" or "POST". Default: "GET"
url  Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key  A privileged CKAN API key. Default: your key set with ckanr_setup
as  (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(. . . , simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
...  Curl args passed on to verb-POST (optional)

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/",
  key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("foobarrrr", author="Jane Doe"))

# then create a resource
file <- system.file("examples", "actinidiaceae.csv", package = "ckanr")
(xx <- resource_create(package_id = res$id,
  description = "my resource",
  name = "bears",
  upload = file,
  extras = list(species = "grizzly"),
  rcurl = "http://google.com")
)```
resource_delete

Delete a resource.

Description

Delete a resource.

Usage

resource_delete(id, url = get_default_url(), key = get_default_key(), ...)

Arguments

id (character) Resource identifier.
url Base url to use. Default: https://data.ontario.ca See also ckanr_setup() and get_default_url().
key A privileged CKAN API key. Default: your key set with ckanr_setup
... Curl args passed on to crul::verb-POST (optional)

Examples

## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = Sys.getenv("CKAN_DEMO_KEY"))

# create a package
(res <- package_create("yellow9"))

# then create a resource
file <- system.file("examples", "actinidiaceae.csv", package = "ckanr")
(xx <- resource_create(res,
    description = "my resource",
    name = "bears",
    upload = file,
    rcurl = "http://google.com"
))

# delete the resource
resource_delete(xx)
resource_patch

Update a resource’s metadata

**Description**

Update a resource’s metadata

**Usage**

```r
resource_patch(
  x,
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

**Arguments**

- `x` (list) A list with key-value pairs
- `id` (character) Resource ID to update (required)
- `url` Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- `key` A privileged CKAN API key. Default: your key set with ckanr_setup
- `as` (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
- `...` Curl args passed on to verb-POST (optional)

**Examples**

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org", key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("twist", author="Alexandria"))

# then create a resource
file <- system.file("examples", "actinidiaceae.csv", package = "ckanr")
(xx <- resource_create(package_id = res$id, description = "my resource"))
```
# Get a resource
res <- resource_show(xx$id)
res$description

# Make some changes
x <- list(description = "My newer description")
z <- resource_patch(x, id = res)
z$description

# Add an extra key:value pair
extra <- list("extra_key" = "my special value")
zz <- resource_patch(extra, id = res)
zz$extra_key

## End(Not run)

### resource_search

Search for resources.

#### Description

Search for resources.

#### Usage

resource_search(
    q,
    sort = NULL,
    offset = NULL,
    limit = NULL,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

#### Arguments

- **q**  
  Query terms. It is a string of the form `field:term` or a vector/list of strings, each of the same form. Within each string, `field` is a field or extra field on the Resource domain object. If `field` is hash, then an attempt is made to match the term as a prefix of the Resource.hash field. If `field` is an extra field, then an attempt is made to match against the extra fields stored against the Resource.

- **sort**  
  Field to sort on. You can specify ascending (e.g., score desc) or descending (e.g., score asc), sort by two fields (e.g., score desc, price asc), or sort by a function (e.g., sum(x_f, y_f) desc, which sorts by the sum of x_f and y_f in a descending order).

- **offset**  
  Record to start at, default to beginning.
resource_show

Show a resource.

Description

Show a resource.

Usage

resource_show(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

id (character) Resource identifier.
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup.
resource_update

Update a resource’s file attachment

Description

This function will only update a resource’s file attachment and the metadata key "last_updated". Other metadata, such as name or description, are not updated.

The new file must exist on a local path. R objects have to be written to a file, e.g. using `tempfile()` - see example.

For convenience, CKAN base url and API key default to the global options, which are set by `ckanr_setup`.

---

**as**  
(character) One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)

...  
Curl args passed on to verb-POST (optional)

Examples

## Not run:

```r
# Setup
ckanr_setup(url = "https://demo.ckan.org/",
key = Sys.getenv("CKAN_DEMO_KEY"))

# create a package
(res <- package_create("yellow7"))

# then create a resource
file <- system.file("examples", "actinidiaceae.csv", package = "ckanr")
(xx <- resource_create(package_id = res$id,
                        description = "my resource",
                        name = "bears",
                        upload = file,
                        rcurl = "http://google.com")

# show the resource
resource_show(xx$id)

# eg. from the NHM CKAN store
resource_show(id = "05ff2255-c38a-40c9-b657-4ccb55ab2feb",
              url = "http://data.nhm.ac.uk")
```

## End(Not run)
resource_update

Usage

resource_update(
  id,
  path,
  extras = list(),
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

id (character) Resource ID to update (required)
path (character) Local path of the file to upload (required)
extras (list) - the resources’ extra metadata fields (optional)
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb-POST (optional)

Value

The HTTP response from CKAN, formatted as list (default), table, or JSON.

References

http://docs.ckan.org/en/latest/api/index.html#ckan.logic.action.create.resource_create

Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# Get file
path <- system.file("examples", "actinidiaceae.csv", package = "ckanr")

# Create package, then a resource within that package
(res <- package_create("newpackage10"))
(xx <- resource_create(package_id = res$id,
  description = "my resource",
  name = "bears",
  upload = path,
  rcurl = "http://google.com")
## Modify dataset, here lowercase strings in one column

```r
dat <- read.csv(path, stringsAsFactors = FALSE)
dat$Family <- tolower(dat$Family)
newpath <- tempfile(fileext = "csv")
write.csv(dat, file = newpath, row.names = FALSE)
```

# Upload modified dataset

```r
## Directly from output of resource_create
resource_update(xx, path=newpath)

## or from the resource id
resource_update(xx$id, path=newpath)

## optionally include extra tags
resource_update(xx$id, path=newpath,
               extras = list(some="metadata"))
```

### Using default settings

```r
ckanr_setup(url = "http://demo.ckan.org/", key = "my-demo-ckan-org-api-key")
path <- system.file("examples", "actinidiaceae.csv", package = "ckanr")
resource_update(id="an-existing-resource-id", path = path)
```

### Using an R object written to a tempfile, and implicit CKAN URL and API key

```r
write.csv(data <- installed.packages(), path <- tempfile(fileext = "csv"))
ckanr_setup(url = "http://demo.ckan.org/", key = "my-demo-ckan-org-api-key")
resource_update(id="an-existing-resource-id", path = path)
```

### Testing: see `ckanr_setup` to set default test CKAN url, key, package id

```r
ckanr_setup(test_url = "http://my.ckan.org/",
            test_key = "my-ckan-api-key",
            test_did = "an-existing-package-id",
            test_rid = "an-existing-resource-id")
resource_update(id = get_test_rid(),
                path = system.file("examples",
                                   "actinidiaceae.csv",
                                   package = "ckanr"),
                key = get_test_key(),
                url = get_test_url())
```

### other file formats

```r
## html
path <- system.file("examples", "mapbox.html", package = "ckanr")
```

### Create package, then a resource within that package

```r
(res <- package_create("mappkg"))
(xx <- resource_create(package_id = res$id,
                        description = "a map, yay",
                        name = "mappyay",
                        upload = path,
                        rcurl = "http://google.com")
```
browseURL(xx$url)

# Modify dataset, here lowercase strings in one column
dat <- readLines(path)
dat <- sub("-111.06", "-115.06", dat)
newpath <- tempfile(fileext = ".html")
cat(dat, file = newpath, sep = "\n")

# Upload modified dataset
## Directly from output of resource_create
(xxx <- resource_update(xx, path=newpath))
browseURL(xxx$url)

## End(Not run)

---

**revision_list**

*Return a list of the IDs of the site’s revisions.*

### Description

Return a list of the IDs of the site’s revisions.

### Usage

```r
revision_list(
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

### Arguments

- **url** Base url to use. Default: https://data.ontario.ca/ See also `ckanr_setup` and `get_default_url`.
- **key** A privileged CKAN API key, Default: your key set with `ckanr_setup`.
- **as** (character) One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
- **...** Curl args passed on to `verb=POST` (optional)
Examples

## Not run:
revision_list()
revision_list(as = "table")
revision_list(as = "json")

## End(Not run)

servers

CKAN server URLs and other info

Description

CKAN server URLs and other info

Usage

servers()

Details

Comes from the links at https://ckan.org/about/instances/

There were a number of other URLs for CKAN instances in the CKAN URL above, but some sites are now gone completely, or if they do exist, I can’t figure out how to get access to the CKAN API on their instance.

Examples

## Not run:
servers()
ckan_info(servers()[5])

# what version is each CKAN server running
out <- lapply(servers()[1:6], function(w) {
  cat(w, sep='\n')
  ckan_info(w)
})
vapply(out, "[[", ",", "ckan_version")

## End(Not run)
**src_ckan**  
*Connect to CKAN with dplyr*

**Description**

Use `src_ckan` to connect to an existing CKAN instance and `tbl` to connect to tables within that CKAN based on the DataStore Data API.

**Usage**

```r
src_ckan(url)
```

**Arguments**

- `url`, the url of the CKAN instance

**Examples**

```r
## Not run:
library("dplyr")

# To connect to a CKAN instance first create a src:
my_ckan <- src_ckan("http://demo.ckan.org")

# List all tables in the CKAN instance
db_list_tables(my_ckan$con)

# Then reference a tbl within that src
my_tbl <- tbl(src = my_ckan, name = "44d7de5f-7029-4f3a-a812-d7a70895da7d")

# You can use the dplyr verbs with my_tbl. For example:
dplyr::filter(my_tbl, GABARITO == "C")

## End(Not run)
```

**tag_create**  
*Create a tag*

**Description**

IMPORTANT: You must be a sysadmin to create vocabulary tags.
tag_list

Usage

tag_create(
    name,
    vocabulary_id,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

Arguments

name (character) The name for the new tag, a string between 2 and 100 characters long containing only alphanumeric characters and - _, and ., e.g. 'Jazz'

vocabulary_id (character) The id of the vocabulary that the new tag should be added to, e.g. the id of vocabulary 'Genre'

url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.

key A privileged CKAN API key, Default: your key set with ckanr_setup

as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)

... Curl args passed on to verb-POST (optional)

Examples

## Not run:
ckanr_setup(url = "https://demo.ckan.org/",
key = Sys.getenv("CKAN_DEMO_KEY"))
tag_create(name = "TestTag1", vocabulary_id = "Testing1")

## End(Not run)

tag_list

List tags.

Description

List tags.
Usage

tag_list(
    query = NULL,
    vocabulary_id = NULL,
    all_fields = FALSE,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

Arguments

query (character) A tag name query to search for, if given only tags whose names contain this string will be returned

vocabulary_id (character) The id or name of a vocabulary, if given, only tags that belong to this vocabulary will be returned

all_fields (logical) Return full tag dictionaries instead of just names. Default: FALSE

url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.

key A privileged CKAN API key, Default: your key set with ckanr_setup

as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)

... Curl args passed on to verb-POST (optional)

Examples

## Not run:
# list all tags
tag_list()

# search for a specific tag
tag_list(query = 'aviation')

# all fields
tag_list(all_fields = TRUE)

# give back different data formats
tag_list('aviation', as = 'json')
tag_list('aviation', as = 'table')

## End(Not run)
**tag_search**

*Search tags.*

**Description**

Search tags.

**Usage**

```r
tag_search(
  query = NULL,
  vocabulary_id = NULL,
  offset = 0,
  limit = 31,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

**Arguments**

- **query** (character) A tag name query to search for, if given only tags whose names contain this string will be returned; one or more search strings
- **vocabulary_id** (character) The id or name of a vocabulary, if give only tags that belong to this vocabulary will be returned
- **offset** (numeric) Where to start getting activity items from (optional, default: 0)
- **limit** (numeric) The maximum number of activities to return (optional, default: 31)
- **url** Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- **key** A privileged CKAN API key, Default: your key set with ckanr_setup
- **as** (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
- **...** Curl args passed on to verb-POST (optional)

**Examples**

```r
## Not run:
tag_search(query = 'ta')
tag_search(query = c('ta', 'al'))
# different formats back
tag_search(query = 'ta', as = 'json')
```
tag_show

tag_search(query = 'ta', as = 'table')

## End(Not run)

tag_show Show a tag.

### Description

Show a tag.

### Usage

tag_show(
    id,
    include_datasets = FALSE,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

### Arguments

- **id** (character) The name or id of the tag
- **include_datasets** include a list of up to 1000 of the tag’s datasets. Limit 1000 datasets, use `package_search()` for more. (optional, default: FALSE)
- **url** Base url to use. Default: https://data.ontario.ca/ See also `ckanr_setup` and `get_default_url`.
- **key** A privileged CKAN API key, Default: your key set with `ckanr_setup`
- **as** (character) One of list (default), table, or json. Parsing with table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
- **...** Curl args passed on to `verb-POST` (optional)

### Examples

```r
## Not run:
# get tags with tag_list()
tag_list()
tags <- tag_list()
tag[[30]]$id

# show a tag
(x <- tag_show(tags[[30]]$id))
```
# give back different data formats
tag_show(tags[[30]]$id, as = 'json')
tag_show(tags[[30]]$id, as = 'table')

## End(Not run)

user_activity_list Return a list of a user's activities

Description

Return a list of a user's activities

Usage

user_activity_list(
  id,
  offset = 0,
  limit = 31,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

id (character) User identifier.
offset (numeric) Where to start getting activity items from (optional, default: 0)
limit (numeric) The maximum number of activities to return (optional, default: 31)
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
...

Curl args passed on to verb-POST (optional)

Examples

## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/",
key = getOption("ckan_demo_key"))
user_create

Create a user.

Description

Create a user.

Usage

```r
user_create(
  name,  # (character) the name of the new user, a string between 2 and 100 characters in length, containing only lowercase alphanumeric characters, - and _ (required)
  email,  # (character) the email address for the new user (required)
  password,  # (character) the password of the new user, a string of at least 4 characters (required)
  id = NULL,  # (character) the id of the new user (optional)
  fullname = NULL,  # (character) user full name
  about = NULL,  # (character) a description of the new user (optional)
  openid = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments
user_delete

Delete a user.

Description

Delete a user.

Usage

```r
user_delete(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

References

http://docs.ckan.org/en/latest/api/index.html#ckan.logic.action.create.user_create

Examples

```r
## Not run:
# Setup
  key = "824e7c50-9577-4bfa-bf32-246ebed1a8a2")

# create a user
user_create(name = 'stacy', email = "stacy@aaaa.com",
  password = "helloworld")

## End(Not run)
```
Arguments

id (character) the id of the new user (required)
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb-POST (optional)

References

http://docs.ckan.org/en/latest/api/index.html#ckan.logic.action.delete.user_delete

Examples

```r
## Not run:
# Setup
key = "824e7c50-9577-4bfa-bf32-246ebed1a8a2")

# create a user
res <- user_delete(name = 'stacy', email = "stacy@aaaa.com",
password = "helloworld")

# then, delete a user
user_delete(id = "stacy")

## End(Not run)
```

user_followee_count 

Return a a user’s follower count

Description

Return a a user’s follower count

Usage

```r
user_followee_count(

id,
url = get_default_url(),
key = get_default_key(),
as = "list",
...
)
```
user_follower_count

Arguments

id (character) User identifier.
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame’s when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb-POST (optional)

Examples

## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# list package activity
user_followee_count('sckottie')

# input a ckan_user object
(x <- user_show('sckottie'))
user_followee_count(x)

# output different data formats
user_followee_count(x, as = "table")
user_followee_count(x, as = "json")

## End(Not run)

user_follower_count Return a user's follower count

Description

Return a user's follower count

Usage

user_follower_count(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

Arguments

- **id**: (character) User identifier.
- **url**: Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
- **key**: A privileged CKAN API key. Default: your key set with ckanr_setup
- **as**: (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
- **...**: Curl args passed on to verb-POST (optional)

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/*", key = getOption("ckan_demo_key"))

# list package activity
user_follower_count('sckottie')

# input a ckan_user object
(x <- user_show('sckottie'))
user_follower_count(x)

# output different data formats
user_follower_count(x, as = "table")
user_follower_count(x, as = "json")
```

user_follower_list

Return a a user's follower count

Description

Return a a user's follower count

Usage

```r
user_follower_list(
  id,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```
Arguments

id (character) User identifier.
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses
jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to
parse data to data.frame’s when possible, so the result can vary from a vector,
list or data.frame. (required)
...
Curl args passed on to verb-POST (optional)

Examples

```r
## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# list package activity
user_follower_list('scottie')

# input a ckan_user object
(x <- user_show('scottie'))
user_follower_list(x)

# output different data formats
user_follower_list(x, as = "table")
user_follower_list(x, as = "json")

## End(Not run)
```

user_list Return a list of the site’s user accounts.

Description

Return a list of the site’s user accounts.

Usage

```r
user_list(
  q = NULL,
  order_by = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```
Arguments

q (character) Restrict the users returned to those whose names contain a string
order_by (character) Which field to sort the list by (optional, default: 'name')
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key, Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb=POST (optional)

Examples

```r
## Not run:
# all users
user_list()

# search for a user
user_list(q = "j")

# different data formats
user_list(q = "j", as = "table")
user_list(q = "j", as = "json")

## End(Not run)
```

user_show Show a user.

Description

Show a user.

Usage

```r
user_show(
  id,
  user_obj = NULL,
  include_datasets = FALSE,
  include_num_followers = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```
Arguments

id (character) Package identifier.
user_obj (user dictionary) The user dictionary of the user (optional)
include_datasets (logical) Include a list of datasets the user has created. If it is the same user or a sysadmin requesting, it includes datasets that are draft or private. (optional, default:False, limit:50)
include_num_followers (logical) Include the number of followers the user has (optional, default:False)
url Base url to use. Default: https://data.ontario.ca/ See also ckanr_setup and get_default_url.
key A privileged CKAN API key. Default: your key set with ckanr_setup
as (character) One of list (default), table, or json. Parsing with table option uses jsonlite::fromJSON(..., simplifyDataFrame = TRUE), which attempts to parse data to data.frame's when possible, so the result can vary from a vector, list or data.frame. (required)
... Curl args passed on to verb=POST (optional)

Examples

## Not run:
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# show user
user_show('sckottie')

# include datasets
user_show('sckottie', include_datasets = TRUE)

# include datasets
user_show('sckottie', include_num_followers = TRUE)

## End(Not run)
Index

* ckanr settings
  ckanr_settings, 11
* package
  ckanr-package, 3
  as.ckan_group, 5
  as.ckan_organization, 5
  as.ckan_package, 6
  as.ckan_related, 7
  as.ckan_resource, 8
  as.ckan_tag, 9
  as.ckan_user, 9

changes, 10
ckan_classes, 14
ckan_fetch, 15
ckan_info, 17
ckan_version (ckan_info), 17
ckan_version(), 49
ckanr (ckanr-package), 3
ckanr_deprecated, 11
ckanr-package, 3
ckanr_settings, 11
ckanr_setup, 10, 11, 12, 18, 19, 21, 22,
  24–30, 32–39, 42–47, 49, 50, 52–56,
  58, 60–62, 64, 66, 68, 71–75, 77–83
ckanr_setup(), 3, 12, 13, 18, 27, 43, 55, 61
crul::proxy(), 13
crul::verb-GET, 18
crul::verb-POST, 27, 43, 55, 61
dashboard_activity_list, 18
dashboard_count, 19
dplyr-interface (src_ckan), 70
ds_create, 20
ds_create(), 4
ds_create_dataset, 21
ds_create_dataset(), 4, 11
ds_search, 23
ds_search(), 4
ds_search_sql, 24
ds_search_sql(), 4

fromJSON, 15
get_default_key (ckanr_settings), 11
get_default_key(), 3, 12
get_default_url, 10, 18, 19, 21, 22, 24–26,
  28–30, 32–39, 42, 44–47, 49, 50,
  52–54, 56, 58, 60, 62, 64, 66, 68,
  71–75, 77–83
get_default_url (ckanr_settings), 11
get_default_url(), 12, 18, 27, 43, 55, 61
get_test_behaviour (ckanr_settings), 11
get_test_did (ckanr_settings), 11
get_test_did(), 12
get_test_gid (ckanr_settings), 11
get_test_gid(), 12
group_create, 25
group_delete, 27
group_list, 28
group_list(), 4, 26, 42
group_patch, 29
group_show, 30
group_show(), 5
group_update, 31

is.ckan_group (as.ckan_group), 5
is.ckan_organization
  (as.ckan_organization), 5
is.ckan_package (as.ckan_package), 6
is.ckan_related (as.ckan_related), 7
is.ckan_resource (as.ckan_resource), 8
is.ckan_tag (as.ckan_tag), 9
is.ckan_user (as.ckan_user), 9

license_list, 32

organization_create, 33
organization_create(), 4
organization_delete, 35
organization_list, 36
organization_list(), 4, 42
organization_purge, 37
organization_show, 38
organization_show(), 4, 6

package_activity_list, 39
package_create, 40
package_create(), 4
package_delete, 42
package_list, 43
package_list_current, 44
package_patch, 45
package_revision_list, 46
package_search, 48
package_search(), 74
package_show, 50
package_show(), 6
package_update, 51
ping, 52

read.csv, 15
read.csv(), 4
read.table, 15
read_excel, 15
read_html, 15
read_xml, 15
related_create, 53
related_delete, 54
related_list, 55
related_show, 56
related_show(), 7
related_update, 57
resource_create, 59
resource_create(), 4, 11, 20, 41
resource_delete, 61
resource_patch, 62
resource_search, 63
resource_show, 64
resource_show(), 8
resource_update, 65

revision_list, 68
servers, 69
servers(), 4
src_ckan, 70
st_read, 15
tag_create, 70
tag_create(), 41
tag_list, 71
tag_list(), 4
tag_search, 73
tag_show, 74
tag_show(), 9

user_activity_list, 75
user_create, 76
user_delete, 77
user_followee_count, 78
user_follower_count, 79
user_follower_list, 80
user_list, 81
user_list(), 4
user_show, 82
user_show(), 10