Package ‘ckanr’

October 11, 2019

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 0.4.0

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

VignetteBuilder knitr

Encoding UTF-8

Depends DBI (>= 0.3.1)

Imports methods, stats, utils, httr (>= 1.0.0), jsonlite (>= 0.9.17), dplyr (>= 0.7.0), dbplyr, magrittr

Suggests knitr, rmarkdown, sf, readxl, testthat, xml2, lazyeval

RoxygenNote 6.1.1

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 2019-10-11 21:00:02 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 ........................................................ 8
as.ckan_user ..................................................... 9
changes ........................................................... 10
ckanr-deprecated ............................................... 11
ckanr_settings ................................................. 11
ckanr_setup .................................................... 12
ckanr_classes ................................................... 13
ckan_fetch ...................................................... 14
ckan_info ........................................................ 16
dashboard_activity_list ...................................... 17
dashboard_count ............................................... 18
ds_create ........................................................ 19
ds_create_dataset ............................................. 20
ds_search ........................................................ 21
ds_search_sql ................................................... 22
group_create .................................................... 23
group_delete .................................................... 25
group_list ........................................................ 26
group_patch ..................................................... 27
group_show ...................................................... 28
group_update .................................................... 29
license_list ..................................................... 30
organization_create ........................................... 30
organization_delete .......................................... 32
organization_list .............................................. 33
organization_show ............................................ 34
package_activity_list ........................................ 35
package_create ................................................ 36
package_delete ................................................ 38
package_list ................................................... 38
package_list_current ......................................... 39
package_patch .................................................. 40
package_revision_list ........................................ 41
package_search ............................................... 42
package_show ................................................... 44
package_update ................................................. 45
ping .............................................................. 46
related_create ................................................. 47
related_delete ............................................... 48
related_list .................................................... 49
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 http://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 ([http://docs.ckan.org/en/latest/maintaining/datastore.html](http://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

Description

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

Description

ckan_organization class helpers

Usage

as.ckan_organization(x, ...)

is.ckan_organization(x)
as.ckan_package

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"))

(ogs <- organization_list())
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)

as.ckan_package ckan_package class helpers

Description

ckan_package class helpers

Usage

as.ckan_package(x, ...)

is.ckan_package(x)

Arguments

x Variety of things, character, list, or ckan_package class object

Further args passed on to package_show if character given

Examples

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

(pkgs <- package_search())
pkgs$results
pkgs$results[[3]]
as.ckan_related

# create item class from only an item ID
as.ckan_package("0699f475-6978-473a-8448-42585074b6f1")

# gives back itself
(x <- as.ckan_package("0699f475-6978-473a-8448-42585074b6f1"))
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("foobbbbbarrrr") %>%
  related_create(title = "my resource",
                 type = "visualization"))

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

# gives back itself
(x <- as.ckan_related(x$id))
as.ckan_related(x)

## End(Not run)
as.ckan_resource  ckan_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:mydataset'))
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  ckan_tag class helpers

Description

ckan_tag class helpers

Usage

as.ckan_tag(x, ...)

is.ckan_tag(x)
as.ckan_user

Arguments

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

Further args passed on to tag_show if character given

Examples

## 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 ckan_user class helpers

Description

ckan_user class helpers

Usage

as.ckan_user(x, ...)

is.ckan_user(x)

Arguments

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

Further args passed on to user_show if character given

Examples

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

(usrs <- user_list())
usrs[1:3]
usrs[[3]]
# create item class from only an item ID
as.ckan_user(usrs[[3]]$id)

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

## End(Not run)

### 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**

```r
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: `http://data.techno-science.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 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

    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()
12

**ckanr_setup**

**Value**

`ckanr_setup` 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**

```r
ckanr_settings()
```

---

**ckanr_setup**

Configure default CKAN settings

**Description**

Configure default CKAN settings

**Usage**

```r
ckanr_setup(url = "http://data.techno-science.ca/", key = NULL,
             test_url = NULL, test_key = NULL, test_did = NULL,
             test_rid = NULL, test_gid = NULL, test_oid = NULL,
             test_behaviour = NULL, proxy = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>url</code></td>
<td>A CKAN URL (optional), default: &quot;<a href="http://data.techno-science.ca/">http://data.techno-science.ca/</a>&quot;</td>
</tr>
<tr>
<td><code>key</code></td>
<td>A CKAN API key (optional, character)</td>
</tr>
<tr>
<td><code>test_url</code></td>
<td>(optional, character) A valid CKAN URL for testing purposes</td>
</tr>
<tr>
<td><code>test_key</code></td>
<td>(optional, character) A valid CKAN API key privileged to create datasets at test_url</td>
</tr>
<tr>
<td><code>test_did</code></td>
<td>(optional, character) A valid CKAN dataset ID, existing at test_url</td>
</tr>
<tr>
<td><code>test_rid</code></td>
<td>(optional, character) A valid CKAN resource ID, attached to did</td>
</tr>
<tr>
<td><code>test_gid</code></td>
<td>(optional, character) A valid CKAN group name at test_url</td>
</tr>
<tr>
<td><code>test_oid</code></td>
<td>(optional, character) A valid CKAN organization name at test_url</td>
</tr>
<tr>
<td><code>test_behaviour</code></td>
<td>(optional, character) Whether to fail (&quot;FAIL&quot;) or skip (&quot;SKIP&quot;) writing tests in case of problems with the configured test CKAN.</td>
</tr>
<tr>
<td><code>proxy</code></td>
<td>an object of class <code>request</code> from a call to <code>httr::use_proxy</code></td>
</tr>
</tbody>
</table>
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

```r
# CKAN users without admin/editor privileges could run:
ckanr_setup(url = "http://data.techno-science.ca/")

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

# ckanR developers/testers can run:
ckanr_setup(url = "http://data.techno-science.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")

# Not specifying the default CKAN URL will reset the CKAN URL to its default
# "http://data.techno-science.ca/":
ckanr_setup()

# set a proxy
ckanr_setup(proxy = httr::use_proxy("64.251.21.73", 8080))
ckanr_settings()
## run without setting proxy to reset to no proxy
ckanr_setup()
ckanr_settings()
```

ckan S3 classes

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 a 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 a 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

ckan_fetch(x, store = "session", path = "file", format = NULL, key = get_default_key(), ...)

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)
- ...: Curl arguments passed on to GET
ckan_fetch

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()
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")
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"])

## 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: `http://data.techno-science.ca`. See also `ckanr_setup` and `get_default_url`. (required)
- **...** Curl args passed on to `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

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: `http://data.techno-science.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 `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)
```
dashboard_count  

Number of new activities of an authorized user

Description

Number of new activities of an authorized user

Usage

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

Arguments

url  
Base url to use. Default: http://data.techno-science.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 POST (optional)

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

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

resource_id  (string) Resource id that the data is going to be stored against.
resource  (dictionary) Resource dictionary that is passed to resource_create(). Use instead of resource_id (optional)
force  (logical) Set to TRUE to edit a read-only resource. Default: FALSE
aliases  (character) Names for read only aliases of the resource. (optional)
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: http://data.techno-science.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 POST (optional)

References

ds_create_dataset

Datastore - create a new resource on an existing dataset

Description

Datastore - create a new resource on an existing 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: http://data.techno-science.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 POST (optional)

Details

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

References

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

Examples

```r
## Not run:
path <- system.file("examples", "actinidiaceae.csv", package = "ckanr")
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)
url Base url to use. Default: http://data.techno-science.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 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

```r
## 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
group_create

Usage

ds_search_sql(sql, url = get_default_url(), key = get_default_key(),
   as = "list", ...)

Arguments

  sql (character) A single SQL select statement. (required)
  url Base url to use. Default: http://data.techno-science.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 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", ...)

## 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)
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'

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 (list of dictionaries) The datasets (packages) that belong to the group, 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)

groups (list of dictionaries) The groups to which the dataset belongs (optional), each group dictionary should have one or more of the following keys 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: http://data.techno-science.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

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

Create a group

(res <- group_create("fruitloops2", description="A group about fruitloops"))
res$user
res$num_followers

## End(Not run)

---

### group_delete

**Delete a group**

**Description**

Delete a group

**Usage**

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

## End(Not run)
```

**Arguments**

- `id` (character) The id of the group. Required.
- `url` Base url to use. Default: [http://data.techno-science.ca](http://data.techno-science.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 POST (optional)

**Examples**

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

# create a group
(res <- group_create("lions", description="A group about lions"))

# show the group
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: http://data.techno-science.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 POST (optional)

Examples

```r
## Not run:
group_list()
group_list(as = 'json')
group_list(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: http://data.techno-science.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 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

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: http://data.techno-science.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 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:
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
group_show(res[[1]]$name, as = 'json')
group_show(res[[1]]$name, as = 'table')
```

## End(Not run)
group_update

Update a group

Description

Update a group

Usage

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

Arguments

- **x** (list) A list with key-value pairs
- **id** (character) Package identifier
- **url** Base url to use. Default: http://data.techno-science.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 POST (optional)

Examples

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

# First, create a group
grp <- group_create("water-bears2")
group_show(grp)

## update just chosen things
# Make some changes
x <- list(description = "A group about water bears and people that love them")

# Then update the package
group_update(x, id = grp)
```

## 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

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

Arguments

- **id** (character) Package identifier.
- **url** Base url to use. Default: `http://data.techno-science.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 `POST` (optional)

Examples

```r
## 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

```r
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 the id of the organization (optional)

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: http://data.techno-science.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 POST (optional)

Examples

## 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: http://data.techno-science.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 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: http://data.techno-science.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 POST (optional)

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

# list organizations
res <- organization_list()
res[1:2]

# Different data formats
organization_list(as = 'json')
organization_list(as = 'table')

## 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: `http://data.techno-science.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 `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:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

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

## End(Not run)
```
package_activity_list  

Return a list of the package’s activity

Description

Return a list of the package’s activity

Usage

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: http://data.techno-science.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 POST (optional)

Examples

## 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_create

Create a package

Description

Create a package

Usage

```r
package_create(name = NULL, title = NULL, 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,
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(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’</td>
</tr>
<tr>
<td>title</td>
<td>(character) the title of the dataset (optional, default: same as name)</td>
</tr>
<tr>
<td>author</td>
<td>(character) the name of the dataset’s author (optional)</td>
</tr>
<tr>
<td>author_email</td>
<td>(character) the email address of the dataset’s author (optional)</td>
</tr>
<tr>
<td>maintainer</td>
<td>(character) the name of the dataset’s maintainer (optional)</td>
</tr>
<tr>
<td>maintainer_email</td>
<td>(character) the email address of the dataset’s maintainer (optional)</td>
</tr>
<tr>
<td>license_id</td>
<td>(license id string) - the id of the dataset’s license, see license_list() for available values (optional)</td>
</tr>
<tr>
<td>notes</td>
<td>(character) a description of the dataset (optional)</td>
</tr>
<tr>
<td>package_url</td>
<td>(character) a URL for the dataset’s source (optional)</td>
</tr>
<tr>
<td>version</td>
<td>(string, no longer than 100 characters) - (optional)</td>
</tr>
<tr>
<td>state</td>
<td>(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’)</td>
</tr>
<tr>
<td>type</td>
<td>(character) the type of the dataset (optional), IDatasetForm plugins associate themselves with different dataset types and provide custom dataset handling behaviour for these types</td>
</tr>
</tbody>
</table>
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)

relationships_as_object (list of relationship dictionaries) - see package_relationship_create() for the format of relationship dictionaries (optional)

relationships_as_subject (list of relationship dictionaries) - see package_relationship_create() for the format of relationship dictionaries (optional)

groups (list of dictionaries) - the groups to which the dataset belongs (optional), each group dictionary should have one or more of the following keys 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()

owner_org (character) the id of the dataset’s owning organization, see organization_list() or organization_list_for_user() for available values (optional)

url Base url to use. Default: http://data.techno-science.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 POST (optional)

Examples

## Not run:

# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package

## Example 1
(res <- package_create("foobar4", author="Jane Doe"))
res$author

## Example 2 - create package, add a resource
(res <- package_create("helloworld", author="Jane DOe"))

## End(Not run)
package_delete  
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: http://data.techno-science.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 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.
package_list_current

Usage

package_list(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: http://data.techno-science.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 POST (optional)

Examples

## Not run:
package_list()
package_list(as = 'json')
package_list(as = 'table')
package_list(url = 'http://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, 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 (required)
- **key** *(character)* A privileged CKAN API key, Default: your key set with `ckanr_setup`
- **url** *(character)* Base url to use. Default: `http://data.techno-science.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 POST (optional)

### Examples

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

```r
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: `http://data.techno-science.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 `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

## End(Not run)
```
# 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**

```r
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 asc) or descending (e.g., score desc), sort by two fields (e.g., score desc, price asc), or sort by a function (e.g., \(\text{sum}(x_f, y_f)\) 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.
package_search

**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 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: `http://data.techno-science.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 POST (optional)

**Examples**

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

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

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

## End(Not run)
```
package_show

Show a package.

Description
Show a package.

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

Arguments
  id (character) Package identifier.
  use_default_schema (logical) Use default package schema instead of a custom schema defined with an IDatasetForm plugin. Default: FALSE
  url Base url to use. Default: http://data.techno-science.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 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
## 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, url = get_default_url(), key = get_default_key(),
as = "list", ...)

Arguments
x  
(list) A list with key-value pairs
id  
(character) Package identifier
url  
Base url to use. Default: http://data.techno-science.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 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))

## Not run:

## 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**

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

**Arguments**

- `url` Base url to use. Default: `http://data.techno-science.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 `POST` (optional)

**Examples**

```r
## 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.
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: http://data.techno-science.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

## 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")
# related_delete

Delete a related item.

**Description**

Delete a related item.

**Usage**

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

**Arguments**

- **id** (character) Resource identifier.
- **url** Base url to use. Default: [http://data.techno-science.ca](http://data.techno-science.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 `POST` (optional)

**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-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", ...)

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: http://data.techno-science.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 POST (optional)

Examples
## Not run:
related_list()
related_list(as = 'json')
related_list(as = 'table')

## End(Not run)

related_show  Show a related item

Description
Show a related item

Usage
related_show(id, url = get_default_url(), key = get_default_key(),
as = "list", ...)
related_update

Update a related item

Description

Update a related item

Usage

related_update(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 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: `http://data.techno-science.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 POST (optional)

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

Arguments

package_id  (character) id of package that the resource should be added to. This should be
            an alphanumeric string. Required.
rcurl  (character) url of resource. Required.
revision_id  (character) revision id (optional)
description  (character) description (optional). Required.
format  (character) format (optional)
hash  (character) hash (optional)
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)
url  Base url to use. Default: http://data.techno-science.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 POST (optional)
Examples

## Not run:

```r
# Setup
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))

# create a package
(res <- package_create("foobarrer", 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"
))

package_create("foobbbbbbarrrrr") %>%
    resource_create(description = "my resource",
                     name = "bearsareus",
                     upload = file,
                     rcurl = "http://google.com")
```

## End(Not run)

---

resource_delete  

Delete a resource.

Description

Delete a resource.

Usage

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

Arguments

- **id**  
  (character) Resource identifier.

- **url**  
  Base url to use. Default: `http://data.techno-science.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 `POST` (optional)
Examples

```r
## 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)

## End(Not run)
```

---

**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: [http://data.techno-science.ca](http://data.techno-science.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 `POST` (optional)
Examples

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

# Get a resource
res <- resource_show("b859f8b6-f9ea-4392-805e-00511d6cf6c6")
res$description

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

## End(Not run)
```

---

### resource_search

Search for resources.

#### Description

Search for resources.

#### Usage

```r
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 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.
- **limit**: Number of records to return.
- **url**: Base url to use. Default: [http://data.techno-science.ca](http://data.techno-science.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

## Not run:
resource_search(q = 'name:Data')
resource_search(q = 'name:Data', as = 'json')
resource_search(q = 'name:Data', as = 'table')
resource_search(q = 'name:Data', limit = 2, as = 'table')

## End(Not run)

---

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: http://data.techno-science.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

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

# create a package
resource_update

(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**

*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`.

**Usage**

```r
resource_update(id, path, 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)
- `url` Base url to use. Default: `http://data.techno-science.ca`. See also `ckanr_setup` and `get_default_url`.
- `key` A privileged CKAN API key. Default: your key set with `ckanr_setup`
resource_update

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 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

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

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

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

# Create package, then a resource within that package
(res <- package_create("mappkg"))
(xx <- resource_create(package_id = res$id,
                        description = "a map, yay",
                        name = "mapyay",
                        upload = path,
                        rcurl = "http://google.com")
)
browseURL(xx$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: `http://data.techno-science.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 `POST` (optional)

Examples

```r
## 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

```r
servers()
```

Details

Comes from the links at http://ckan.org/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

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

# what version is each CKAN server running
out <- lapply(servers()[1:6], ckan_info)
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.

Usage

```r
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: `http://data.techno-science.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 POST (optional)

Examples

```r
## 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: http://data.techno-science.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 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  

List tags.

Description

List tags.
tag_search

Usage

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
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: http://data.techno-science.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 POST (optional)

Examples

## Not run:
tag_search(query = "Var")

# different formats back
tag_search(query = "ta", as = "json")
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", ...)

---

Usage: `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.
- **vocabulary_id**: (character) The id or name of a vocabulary, if given 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: `http://data.techno-science.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 POST (optional).

Examples:

```r
## Not run:
tag_search(query = "Var")

# different formats back
tag_search(query = "ta", as = "json")
tag_search(query = "ta", as = "table")

## End(Not run)
```

Description:

Show a tag.

Usage: `tag_show(id, include_datasets = FALSE, url = get_default_url(), key = get_default_key(), as = "list", ...)`.
**user_activity_list**

Return a list of a user’s activities

### Description

Return a list of a user’s activities

### Usage

```r
user_activity_list(id, offset = 0, limit = 31,
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>id</td>
<td>(character) User identifier.</td>
</tr>
<tr>
<td>offset</td>
<td>(numeric) Where to start getting activity items from (optional, default: 0)</td>
</tr>
<tr>
<td>limit</td>
<td>(numeric) The maximum number of activities to return (optional, default: 31)</td>
</tr>
<tr>
<td>url</td>
<td>Base url to use. Default: <a href="http://data.techno-science.ca">http://data.techno-science.ca</a>. See also ckanr_setup and get_default_url.</td>
</tr>
<tr>
<td>key</td>
<td>A privileged CKAN API key. Default: your key set with ckanr_setup</td>
</tr>
<tr>
<td>as</td>
<td>(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)</td>
</tr>
<tr>
<td></td>
<td>Curl args passed on to POST (optional)</td>
</tr>
</tbody>
</table>

Examples

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

# list package activity
tactivity_list(‘scottie’)  

# input a ckan_user object  
(x <- user_show(‘scottie’))
tactivity_list(x)  

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

## End(Not run)
```

**user_create**  
Create a user.

**Description**

Create a user.

**Usage**

```r
user_create(name, email, password, id = NULL, fullname = NULL,  
            about = NULL, openid = NULL, url = get_default_url(),  
            key = get_default_key(), as = "list", ...)
```
Delete a user.

Delete a user.

Usage

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

id (character) the id of the new user (required)

url Base url to use. Default: http://data.techno-science.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 POST (optional)

References

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

Examples

## Not run:
# Setup
ckanr_setup(url = "https://data-demo.dpaw.wa.gov.au", key = "824e7c50-9577-4bfa-bf32-246ebeda8a2")

# create a user
res <- user_delete(name = 'stacy', email = "stacy@aaaaa.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

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

id (character) User identifier.

url Base url to use. Default: http://data.techno-science.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 POST (optional)

Examples

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

# list package activity
user_followee_count('scottie')

# input a ckan_user object (x <- user_show('scottie'))
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 a user’s follower count*

**Description**

Return a a user’s follower count

**Usage**

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

Return a a user’s follower count

Description

Return a a user’s follower count

Usage

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

Return a list of the site’s user accounts.

Arguments

id (character) User identifier.
url Base url to use. Default: http://data.techno-science.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 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('sckottie')

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

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

## End(Not run)
```

Description

Return a list of the site’s user accounts.

Usage

`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: http://data.techno-science.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 POST (optional)

Examples

## 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

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: http://data.techno-science.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 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

*Topic 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, 8
as.ckan_user, 9

changes, 10
ckan_classes, 13
ckan_fetch, 14
ckan_info, 16
ckan_version, 43
ckan_version (ckanr_info), 16
ckanr (ckanr-package), 3
ckanr-deprecated, 11
ckanr-package, 3
ckanr_settings, 11
ckanr_setup, 3, 10, 12, 16–20, 22–35, 37–41, 43–57, 60, 62–73
dashboard_activity_list, 17
dashboard_count, 18
dplyr-interface (src_ckan), 61
ds_create, 4, 19
ds_create_dataset, 4, 11, 20
ds_search, 4, 21
ds_search_sql, 4, 22

GET, 14, 16
get_default_key, 3, 12
get_default_key (ckanr_settings), 11
get_default_url, 10, 12, 16–20, 22–35, 37–41, 43–57, 60, 62–73
get_default_url (ckanr_settings), 11
get_test_behaviour (ckanr_settings), 11
get_testdid (ckanr_settings), 11
get_test_gid, 12
get_test_gid (ckanr_settings), 11
get_test_key, 12
get_test_key (ckanr_settings), 11
get_test_oid (ckanr_settings), 11
get_test_rid, 12
get_test_rid (ckanr_settings), 11
get_test_url, 12
get_test_url (ckanr_settings), 11

group_create, 23
group_delete, 25
group_list, 4, 26
group_patch, 27
group_show, 5, 28
group_update, 29

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), 8
is.ckan_user (as.ckan_user), 9

license_list, 30

organization_create, 4, 30
organization_delete, 32
organization_list, 4, 33
organization_show, 4, 6, 34

package_activity_list, 35
package_create, 3, 36
package_delete, 38
package_list, 38
package_list_current, 39
package_patch, 40
package_revision_list, 41
INDEX

package_search, 42, 65
package_show, 6, 44
package_update, 45
ping, 46
read.csv, 4
related_create, 47
related_delete, 48
related_list, 49
related_show, 7, 49
related_update, 50
resource_create, 4, 11, 51
resource_delete, 53
resource_patch, 54
resource_search, 55
resource_show, 8, 56
resource_update, 57
revision_list, 59
servers, 4, 60
src_ckan, 61

tag_create, 62
tag_list, 4, 62
tag_search, 63
tag_show, 9, 64

user_activity_list, 65
user_create, 66
user_delete, 67
user_followee_count, 68
user_follower_count, 69
user_follower_list, 70
user_list, 4, 71
user_show, 9, 72