

Package: ckanr (via r-universe)

October 11, 2026

Title Client for the Comprehensive Knowledge Archive Network ('CKAN')

Description Client for the Comprehensive Knowledge Archive Network ('CKAN') application programming interface ('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' extension.

Version 0.9.1

License MIT + file LICENSE

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

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

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0), DBI (>= 0.3.1)

Imports methods, stats, utils, crul (>= 1.6.0), jsonlite (>= 2.0.0), dplyr (>= 1.2.1), dbplyr (>= 2.6.0)

Suggests arrow (>= 25.0.1), sf (>= 1.1-3), readr (>= 2.2.0), readxl (>= 1.5.0), testthat (>= 3.3.2), xml2 (>= 1.6.0), rlang (>= 1.3.0)

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

X-schema.org-applicationCategory Data Access

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

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libicu-dev libssl-dev

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

Date/Publication 2026-10-11 12:53:04 UTC

RemoteUrl <https://github.com/ropensci/ckanr>

RemoteRef main

RemoteSha 468ba8864a62a7f5e60911a1f48c4b80706cf99e

Contents

ckanr-package	5
activity_helpers	6
activity_purge	9
api_tokens	11
as.ckan_file	12
as.ckan_group	13
as.ckan_organization	14
as.ckan_package	15
as.ckan_related	16
as.ckan_resource	17
as.ckan_resource_view	18
as.ckan_tag	19
as.ckan_user	20
as.ckan_vocabulary	21
bulk_update	21
changes	23
ckan_action	24
ckan_classes	25
ckan_fetch	26
ckan_info	29
ckanr-deprecated	30
ckanr_settings	30
ckanr_setup	31
config_options	33
dashboard_activity_list	35
dashboard_count	36
dataset_purge	37
datastore_extra	37
diagnostics	40
ds_create	41
ds_create_dataset	43
ds_search	45
ds_search_sql	47
files	48
follow_objects	51
follow_users	52
followee_counts	54
followee_lists	55
follower_counts	57
follower_lists	59
format_autocomplete	60
get_site_user	61
group_autocomplete	63
group_create	64
group_delete	66
group_list	67

group_list_authz	68
group_member_create	69
group_member_delete	70
group_package_show	71
group_patch	72
group_purge	73
group_show	75
group_update	76
job_queue	77
license_list	79
member_create	80
member_delete	81
member_list	82
member_roles_list	83
membership	83
organization_autocomplete	84
organization_create	85
organization_delete	87
organization_list	88
organization_list_for_user	89
organization_member_create	90
organization_member_delete	91
organization_patch	92
organization_purge	93
organization_show	94
organization_update	96
package_activity_list	97
package_autocomplete	98
package_collaborator_create	99
package_collaborator_delete	100
package_collaborator_list	101
package_collaborator_list_for_user	102
package_collaborators	103
package_create	103
package_create_default_resource_views	106
package_dataset_extras	107
package_delete	107
package_list	108
package_list_current	109
package_owner_org_update	110
package_patch	111
package_relationship_create	112
package_relationship_delete	113
package_relationship_update	114
package_relationships	115
package_relationships_list	116
package_resource_reorder	117
package_revise	118

package_revision_list	119
package_search	120
package_show	122
package_update	124
ping	125
related_create	126
related_delete	127
related_list	128
related_show	129
resource_create	131
resource_create_default_resource_views	133
resource_delete	134
resource_patch	135
resource_search	137
resource_show	138
resource_update	139
resource_view_clear	143
resource_view_create	144
resource_view_delete	145
resource_view_list	146
resource_view_reorder	147
resource_view_show	148
resource_view_update	149
resource_views	150
revision_list	150
servers	151
src_ckan	152
tag_autocomplete	153
tag_create	154
tag_delete	155
tag_list	156
tag_search	157
tag_show	158
task_status	160
term_translation	162
user_activity_list	163
user_autocomplete	164
user_create	166
user_delete	167
user_invite	168
user_list	169
user_patch	171
user_show	172
user_update	173
vocabulary	175

Description

ckanr is a client for the CKAN API. It wraps all APIs for reading and writing data. If you have problems, or have use cases that the package does not cover yet, get in touch (<https://github.com/ropensci/ckanr/issues>)

CKAN API

Documentation for the CKAN API is at <https://docs.ckan.org/en/2.12/api/index.html>. The package follows the latest 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

The package supports the Datastore extension (<https://docs.ckan.org/en/2.12/maintaining/datastore.html>). It provides these functions:

- [ds_create\(\)](#)
- [ds_create_dataset\(\)](#)
- [ds_search\(\)](#)
- [ds_search_sql\(\)](#)

Fetch

Data comes back in a wide range of formats. The package provides a function to help you fetch metadata. The function also fetches the actual data for a link to a file on a CKAN instance. If you know what you are doing, you can use your preferred tool for the job. For example, you like [read.csv\(\)](#) for reading csv files.

CKAN Instances

A helper function (`servers()`) lists the current CKAN instances that the package knows about. It gives the base URLs that work with this package. The URLs are not necessarily landing pages of each instance, but the URL can 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

Immanuel Costigan <i.costigan@me.com>

Sharla Gelfand

Francisco Alves <fjunior.alves.oliveira@gmail.com>

See Also

Useful links:

- <https://docs.ropensci.org/ckanr/> (website) <https://github.com/ropensci/ckanr> (devel)
- Report bugs at <https://github.com/ropensci/ckanr/issues>

activity_helpers

Activity stream helpers

Description

These helpers wrap CKAN's `ckanext.activity` endpoints. The target CKAN instance must enable the `activity` core plugin.

Usage

```
group_activity_list(  
  id,  
  offset = 0,  
  limit = 31,  
  include_hidden_activity = FALSE,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)  
  
organization_activity_list(  
  id,
```

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

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

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

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

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

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

```

activity_diff(
  id,
  object_type,
  diff_type = "unified",
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

activity_create(
  user_id,
  object_id,
  activity_type,
  data = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

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

```

Arguments

<code>id</code>	(character) Identifier of the target object (group, organization, activity, etc.).
<code>offset</code>	(numeric) Where to start getting activity items from (optional, default: 0)
<code>limit</code>	(numeric) The maximum number of activities to return (optional, default: 31)
<code>include_hidden_activity</code>	(logical) If TRUE, include private activity entries. You must have sysadmin rights.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)
<code>object_type</code>	(character) Domain object that the activity affects, for example "package", "resource", or a plugin-defined type.

diff_type	(character) Diff format that activity_diff() returns, typically "unified".
user_id	(character) User identifier associated with a custom activity_create() entry.
object_id	(character) Target object identifier for activity_create().
activity_type	(character) Activity type string to emit via activity_create().
data	(list character) Optional structured payload that describes the activity body. The function encodes lists as JSON automatically.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the as parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
activity_show(id = "my-activity-id")
group_activity_list(id = "my-group")

## End(Not run)
```

activity_purge	<i>Activity purge helpers (CKAN 2.12+)</i>
----------------	--

Description

These helpers wrap the CKAN 2.12 activity purge feature (<https://github.com/ckan/ckan/pull/8189>). You must have sysadmin rights and the target instance must enable the activity plugin with CKAN 2.12 or later. Each wrapper calls ensure_action_available() so it fails clearly on older instances.

Usage

```
activity_delete(
  id = NULL,
  start_date = NULL,
  end_date = NULL,
  offset_days = NULL,
  keep = NULL,
  batch_size = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

activity_delete_all(
  batch_size = NULL,
```

```

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

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

```

Arguments

<code>id</code>	(character) Activity identifier to delete via <code>activity_delete()</code> . Alternatively provide <code>start_date</code> + <code>end_date</code> or <code>offset_days</code> .
<code>start_date</code>	(character) Start of the deletion range (ISO 8601).
<code>end_date</code>	(character) End of the deletion range (ISO 8601).
<code>offset_days</code>	(numeric) Delete activities older than this many days.
<code>keep</code>	(numeric) Optional. Keep this many most recent activities per item; delete only older ones in the range.
<code>batch_size</code>	(numeric) Optional batch size for large tables.
<code>url</code>	Base URL to use. Default: <code>https://demo.ckan.org/</code> . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
activity_delete_counts()
activity_delete(offset_days = 365)

## End(Not run)

```

Description

Inspect, create, and revoke API tokens for a user.

Usage

```
api_token_list(  
  user_id = NULL,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)  
  
api_token_create(  
  user,  
  name,  
  extra_fields = NULL,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)  
  
api_token_revoke(  
  token = NULL,  
  jti = NULL,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)
```

Arguments

user_id	(character) User identifier to filter tokens.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)

...	Extra curl arguments. The function passes them to verb-POST (optional)
user	(character) Name or id of the token owner.
name	(character) Friendly token name.
extra_fields	(list) Optional named list of additional fields accepted by CKAN or extensions.
token	(character) Encoded API token value.
jti	(character) Token identifier (overrides token when provided).

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-ckan-key")
api_token_list(user_id = "my-user-id")

## End(Not run)
```

as.ckan_file	<i>ckan_file class helpers</i>
--------------	--------------------------------

Description

ckan_file class helpers

Usage

```
as.ckan_file(x, ...)

is.ckan_file(x)
```

Arguments

x	One of character, list, or ckan_file class object
...	Extra arguments. If x is character, the function passes them on to file_show()

Value

A ckan_file object, a list with S3 class ckan_file.

Examples

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

# create item class from only an item ID
as.ckan_file("file-id-here")

# gives back itself
(x <- as.ckan_file(list(id = "file-id-here", name = "file.txt")))
as.ckan_file(x)

## End(Not run)
```

as.ckan_group	<i>ckan_group class helpers</i>
---------------	---------------------------------

Description

ckan_group class helpers

Usage

```
as.ckan_group(x, ...)

is.ckan_group(x)
```

Arguments

x	One of character, list, or ckan_group class object
...	Extra arguments. If x is character, the function passes them on to group_show()

Value

A ckan_group object, a list with S3 class ckan_group.

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)
```

Arguments

x	One of character, list, or ckan_organization class object
...	Extra arguments. If x is character, the function passes them on to organization_show()

Value

A ckan_organization object, a list with S3 class ckan_organization.

Examples

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

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

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

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

## End(Not run)
```

as.ckan_package	<i>ckan_package class helpers</i>
-----------------	-----------------------------------

Description

ckan_package class helpers

Usage

```
as.ckan_package(x, ...)
```

```
is.ckan_package(x)
```

Arguments

x	One of character, list, or ckan_package class object
...	Extra arguments. If x is character, the function passes them on to package_show() . If GET is not supported, use the <code>http_method</code> parameter to set a different HTTP verb

Value

A ckan_package object, a list with S3 class ckan_package.

Examples

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

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

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

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

## End(Not run)
```

as.ckan_related	<i>ckan_related class helpers</i>
-----------------	-----------------------------------

Description

ckan_related class helpers

Usage

```
as.ckan_related(x, ...)
```

```
is.ckan_related(x)
```

Arguments

x	One of character, list, or ckan_related class object
...	Extra arguments. If x is character, the function passes them on to related_show()

Value

A ckan_related object, a list with S3 class ckan_related.

Examples

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

(x <- package_create("foobbbbbarrrrr") |>
  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	<i>ckan_resource class helpers</i>
------------------	------------------------------------

Description

ckan_resource class helpers

Usage

```
as.ckan_resource(x, ...)
```

```
is.ckan_resource(x)
```

Arguments

x	One of character, list, or ckan_package class object
...	Extra arguments. If x is character, the function passes them on to resource_show()

Value

A ckan_resource object, a list with S3 class ckan_resource.

Examples

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

(resrcs <- resource_search(q = "name:data"))
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_resource_view *ckan_resource_view class helpers*

Description

ckan_resource_view class helpers

Usage

```
as.ckan_resource_view(x, ...)
```

```
is.ckan_resource_view(x)
```

Arguments

x	One of character, list, or ckan_resource_view object
...	Extra arguments. If x is an identifier, the function passes them to resource_view_show() .

Value

A ckan_resource_view object, a list with S3 class ckan_resource_view.

Examples

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

res <- package_show("sample-dataset")
views <- resource_view_list(res$resources[[1]]$id)

# Coerce from ID
as.ckan_resource_view(views[[1]]$id)

# Pass through existing objects
as.ckan_resource_view(views[[1]])

## End(Not run)
```

as.ckan_tag	<i>ckan_tag class helpers</i>
-------------	-------------------------------

Description

ckan_tag class helpers

Usage

```
as.ckan_tag(x, ...)
```

```
is.ckan_tag(x)
```

Arguments

x	One of character, list, or ckan_tag class object
...	Extra arguments. If x is character, the function passes them on to tag_show()

Value

A ckan_tag object, a list with S3 class ckan_tag.

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

<code>x</code>	One of character, list, or ckan_user class object
<code>...</code>	Extra arguments. If <code>x</code> is character, the function passes them on to user_show()

Value

A ckan_user object, a list with S3 class ckan_user.

Examples

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

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

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

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

## End(Not run)
```

as.ckan_vocabulary	<i>ckan_vocabulary class helpers</i>
--------------------	--------------------------------------

Description

ckan_vocabulary class helpers

Usage

```
as.ckan_vocabulary(x, ...)
```

```
is.ckan_vocabulary(x)
```

Arguments

x	One of character, list, or ckan_vocabulary object
...	Extra arguments. If you retrieve by identifier, the function passes them to vocabulary_show().

Value

A ckan_vocabulary object, a list with S3 class ckan_vocabulary.

Examples

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

vocab <- vocabulary_create(name = sprintf("demo_vocab_%s", sample.int(1e4, 1)))

# create class from an id
as.ckan_vocabulary(vocab$id)

# passing through existing objects is a no-op
as.ckan_vocabulary(vocab)

## End(Not run)
```

bulk_update	<i>Bulk update dataset visibility.</i>
-------------	--

Description

Make a list of datasets private, public, or deleted in a single call. You must be authorized to edit the datasets (for example, an editor or admin of the owning organization). See the official API contracts: https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.bulk_update_private, https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.bulk_update_public, https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.bulk_update_delete.

Usage

```

bulk_update_private(
  datasets,
  org_id,
  url = get_default_url(),
  key = get_default_key(),
  ...
)

bulk_update_public(
  datasets,
  org_id,
  url = get_default_url(),
  key = get_default_key(),
  ...
)

bulk_update_delete(
  datasets,
  org_id,
  url = get_default_url(),
  key = get_default_key(),
  ...
)

```

Arguments

datasets	(character or list) The datasets to update: a character vector of dataset ids or names, a list of ckan_package objects, or a list with an id entry per dataset.
org_id	(character or ckan_organization) The id or name of the owning organization, or a ckan_organization object.
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE when CKAN applies the bulk update successfully.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.bulk_update_private

Examples

```

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

```

```

org <- organization_create("bulk-org")
ds1 <- package_create("bulk-dataset-1", owner_org = org$id)
ds2 <- package_create("bulk-dataset-2", owner_org = org$id)

bulk_update_private(c(ds1$id, ds2$id), org_id = org$id)
bulk_update_public(c(ds1$id, ds2$id), org_id = org$id)
bulk_update_delete(c(ds1$id, ds2$id), org_id = org$id)

## 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

```

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

```

Arguments

<code>offset</code>	(numeric) Where to start getting activity items from (optional, default: 0)
<code>limit</code>	(numeric) The maximum number of activities to return (optional, default: 31)
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
changes()
changes(as = "json")
changes(as = "table")

## End(Not run)
```

ckan_action	<i>Generic action function</i>
-------------	--------------------------------

Description

You must set the CKAN action and the HTTP verb. You also set query parameters, body data, and headers.

Usage

```
ckan_action(
  action,
  query = NULL,
  body = NULL,
  headers = list(),
  verb = "POST",
  url = get_default_url(),
  key = get_default_key(),
  ...
)
```

Arguments

action	A valid CKAN API action name (for example, "package_list", "package_show"). See the CKAN API documentation for a full list of actions: https://docs.ckan.org/en/2.12/api/
query	a named list of URL query parameters
body	Data for the body of a request. See https://docs.ropensci.org/crul/reference/verb-POST.html for options
headers	a named list of request headers
verb	HTTP request verb, for example, GET, POST
url	Base url to use. Default: https://demo.ckan.org/ See also ckanr_setup() and get_default_url()
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Curl args. The function sends them to the relevant crul::HttpClient method for the verb parameter (optional)

Value

A text string. The function returns text because the data type is unknown ahead of time. You parse the text as needed.

Examples

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

ckan_action("package_list")
ckan_action("package_list", verb = "GET")
ckan_action("package_list", url = "https://data.nhm.ac.uk")

## End(Not run)
```

ckan_classes

ckanr S3 classes

Description

ckanr S3 classes

The classes

- ckan_package - CKAN package
- ckan_resource - CKAN resource
- ckan_resource_view - CKAN resource view
- ckan_related - CKAN related item
- ckan_file - CKAN file (first-class file entities, CKAN 2.12+)

Coercion

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

Testing for classes

To test the class of an object, use an `is._ckan_*`() function for the classes listed above. The function returns TRUE or FALSE.

Manipulation

These are S3 classes. Each class is an R list with an attached class. The class tells functions what to do with the object. Functions accept flexible inputs and return flexible outputs. You can edit one of these classes yourself by changing values in the list.

See Also

`as.ckan_package()`, `as.ckan_resource()`, `as.ckan_resource_view()`, `as.ckan_related()`, `as.ckan_file()`

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 of the file
store	One of session (default) or disk. The session option stores the file in the R session, and the disk option saves the file to disk.
path	If store="disk", give a path to store the file.
format	Format of the file. If the file URL does not show the format, pass format.
key	A CKAN API key (optional, character)
...	Extra arguments. If store="session", the function passes them to the function that reads the file. See details.

Details

You can pass additional arguments with ... to the function that reads the file. The function uses ... only when store="session" and the file is *not* a ZIP file.

This list shows the function that reads each file format. You can see what additional arguments are available:

- csv: [read.csv](#)
- xls, xlsx: [read_excel](#)
- xml: [read_xml](#)
- html: [read_html](#)
- json: [fromJSON](#)
- shp, geojson: [st_read](#)
- txt: [read.table](#)
- parquet: [read_parquet](#)

Value

With store = "session" (default), the file contents, usually a data.frame (Excel files with multiple sheets and ZIP files return a named list). With store = "disk", a list with download details including the file path.

Examples

```
## Not run:
# CSV file
ckanr_setup("https://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("https://datamx.io")
res <- resource_show(
  id = "e883510e-a082-435c-872a-c5b915857ae1",
  as = "table"
)
head(ckan_fetch(res$url))

# Excel file, multiple sheets - requires readxl package
ckanr_setup()
res <- resource_show(
  id = "ce02a1cf-35f1-41df-91d9-11ed1fdd4186",
  as = "table"
```

```

)
x <- ckan_fetch(res$url)
names(x)
head(x[["Mayor - Maire"]])

# XML file - requires xml2 package
ckanr_setup("https://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("https://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("https://data.surrey.ca")
res <- resource_show(
  id = "8d07c662-800d-4977-9e3e-5a3d2d1e99ab",
  as = "table"
)
head(ckan_fetch(res$url))

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

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

# ZIP file - packages required depends on contents
ckanr_setup("https://ckan0.cf.opendata.inter.prod-toronto.ca")
res <- resource_show(
  id = "bb21e1b8-a466-41c6-8bc3-3c362cb1ed55",

```

```

    as = "table"
  )
  x <- ckan_fetch(res$url)
  names(x)
  head(x[["ChickenpoxAgegroups2017.csv"]])

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

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

# Parquet file - requires arrow package
ckanr_setup("https://data.coat.no")
res <- resource_show(
  id = "b7d1c192-0f44-4bab-bd51-eea0d408a09f",
  as = "table"
)
x <- ckan_fetch(res$url)
head(x)

## End(Not run)

```

ckan_info

Get information on a CKAN server

Description

Get information on a CKAN server

Usage

```
ckan_info(url = get_default_url(), ...)
```

```
ckan_version(url = get_default_url(), ...)
```

Arguments

- `url` Base URL to use. Default: <https://demo.ckan.org/>. See also [ckanr_setup\(\)](#) and [get_default_url\(\)](#). (required)
- `...` Extra curl arguments. The function passes them on to [curl::verb-GET](#) (optional)

Value

For `ckan_info`, the function returns a list with many slots with various info. For `ckan_version`, the function returns a list of length two, with the actual version as character. The second item converts the version to numeric (any dots or letters removed)

Examples

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

ckan_version(servers()[5])

## End(Not run)
```

ckanr-deprecated	<i>Deprecated functions in ckanr</i>
------------------	---

Description

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

Details

- [ds_create_dataset\(\)](#): Another function in this package already provides this functionality. See function [resource_create\(\)](#)

ckanr_settings	<i>Get or set ckanr CKAN settings</i>
----------------	---------------------------------------

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()
```

Value

ckanr_settings prints your base url, API key (if used), and optional test server settings (URL, API key, a dataset ID and a resource ID). ckanr_setup sets your production and test settings. get_test_* gets each of those settings. test_behaviour states whether the CKANR test suite skips ("SKIP") or fails ("FAIL") writing tests when the configured test CKAN settings do not work.

See Also

[ckanr_setup\(\)](#), [get_default_url\(\)](#), [get_default_key\(\)](#), [get_test_url\(\)](#), [get_test_key\(\)](#), [get_test_did\(\)](#), [get_test_rid\(\)](#), [get_test_gid\(\)](#), [get_test_oid](#), [get_test_behaviour](#)

Examples

```
ckanr_settings()
```

ckanr_setup

Configure default CKAN settings

Description

Configure default CKAN settings

Usage

```

ckanr_setup(
  url = "https://demo.ckan.org/",
  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

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

Details

`ckanr_setup()` sets CKAN connection details. If you do not specify a URL or key, the functions use the default URL and API key.

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

The writing tests (create, update, delete) can fail for two reasons. One reason is failures in ckanr code. The tests aim to detect these failures. The other reason is failures in the configured CKAN. These failures are not necessarily a problem with ckanr code, but they stop the tests from proving otherwise.

If you set `test_behaviour` to "SKIP", writing tests skip when the configured test CKAN fails. This helps you test the other functions even when you have no write access to a CKAN instance.

If you set `test_behaviour` to "FAIL", the tester finds problems with both the configured test CKAN and the writing functions.

Value

(invisible) Called for its side effects; sets default and test CKAN settings in environment variables and returns invisibly.

Examples

```
# CKAN users without admin/editor privileges could run:
ckanr_setup(url = "https://demo.ckan.org/")

# Privileged CKAN editor/admin users can run:
ckanr_setup(url = "https://demo.ckan.org/", key = "some-CKAN-API-key")

# ckanR developers/testers can either use the Codespaces/devcontainer test suite
# which automatically sets up all test credentials, or run:
ckanr_setup(
  url = "https://demo.ckan.org/", 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
# "https://demo.ckan.org/":
ckanr_setup()

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

config_options

Runtime configuration helpers

Description

Inspect and update CKAN's runtime-editable configuration keys.

Usage

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

```

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

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

```

Arguments

url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)
option_key	(character) Configuration option identifier.
options	(list) Named list of configuration options to update.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-ckan-key")
config_option_list()
config_option_update(list("ckan.site_title" = "My Portal"))

## 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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## 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	<i>Number of new activities of an authorized user</i>
-----------------	---

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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Details

CKAN does not count activities from the user herself, but they appear in the dashboard. Users do not want notices about things they did themselves.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

# count
dashboard_new_activities_count()

## End(Not run)
```

dataset_purge	<i>Permanently purge a dataset</i>
---------------	------------------------------------

Description

Permanently purge a dataset

Usage

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

Arguments

id	(character or ckan_package) Dataset identifier.
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE when the function purges the dataset successfully.

Examples

```
## Not run:
dataset_purge("dataset-id")

## End(Not run)
```

datastore_extra	<i>Datastore write helpers (CKAN 2.12+ parameters from the start)</i>
-----------------	---

Description

Wrappers for `datastore_upsert`, `datastore_delete`, `datastore_records_delete`, `datastore_info`, `datastore_function_create`, `datastore_function_delete`, and `datastore_run_triggers`. `ds_search()` and `ds_create()` cover read/create; these helpers complete the write/info coverage with the CKAN 2.12 parameters (`include_records`, `include_meta`, `include_fields_schema`, `argmode`) available from the start. See <https://docs.ckan.org/en/2.12/maintaining/datastore.html#the-data-api>.

Usage

```
ds_upsert(  
    resource_id,  
    records = NULL,  
    method = "upsert",  
    force = FALSE,  
    include_records = FALSE,  
    calculate_record_count = "background",  
    dry_run = FALSE,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
ds_delete(  
    resource_id = NULL,  
    filters = NULL,  
    force = FALSE,  
    include_deleted_records = FALSE,  
    calculate_record_count = "background",  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
ds_records_delete(  
    resource_id,  
    filters,  
    force = FALSE,  
    include_deleted_records = FALSE,  
    calculate_record_count = "background",  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
ds_info(  
    resource_id,  
    include_meta = TRUE,  
    include_fields_schema = TRUE,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)
```

```

ds_function_create(
    name,
    rettype = "trigger",
    definition,
    or_replace = FALSE,
    argmode = NULL,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

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

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

```

Arguments

<code>resource_id</code>	(character) Resource id that the data is stored against.
<code>records</code>	(list data.frame) The data, e.g. <code>list(list(a = 1, b = "xyz"))</code> .
<code>method</code>	(character) Upsert method: "upsert" (default), "insert", or "update". upsert/update require a unique key or <code>_id</code> field.
<code>force</code>	(logical) Edit a read-only table. Default: FALSE.
<code>include_records</code>	(logical) If TRUE, return the actual affected rows in the response (CKAN 2.12+). Default: FALSE.
<code>calculate_record_count</code>	(logical character) FALSE skips the count update, TRUE updates immediately, "background" schedules a background job (default).
<code>dry_run</code>	(logical) Abort the transaction instead of committing, e.g. to check validation errors. Default: FALSE.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup

<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to <code>verb-POST</code> (optional)
<code>filters</code>	(list character) Matching conditions to select, e.g. <code>list(name = "fred")</code> . CKAN 2.12+ accepts advanced syntax (ranges, nested AND/OR); the argument passes through untouched.
<code>include_deleted_records</code>	(logical) If TRUE, return the full values of deleted records (CKAN 2.12+). Default: FALSE.
<code>include_meta</code>	(logical) For <code>ds_info()</code> , return table size, index size, row count and aliases (CKAN 2.12+). Default: TRUE.
<code>include_fields_schema</code>	(logical) For <code>ds_info()</code> , return per-field index/unique/notnull status (CKAN 2.12+). Default: TRUE.
<code>name</code>	(character) Trigger function name for <code>ds_function_create()</code> / <code>ds_function_delete()</code> .
<code>rettype</code>	(character) Set to "trigger" (only trigger functions may be created at this time).
<code>definition</code>	(character) PL/pgSQL function body for the trigger.
<code>or_replace</code>	(logical) Replace the function if it exists. Default: FALSE.
<code>argmode</code>	(character) Argument mode for custom DataStore SQL function parameters, e.g. "in", "inout", "out" (CKAN 2.12+, https://github.com/ckan/ckan/pull/8279). Optional.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
ds_info(resource_id = "my-resource-id")
ds_upsert(resource_id = "my-resource-id",
  records = list(list(a = 1, b = "xyz")), method = "upsert")

## End(Not run)
```

diagnostics

Diagnostics helpers

Description

Lightweight wrappers for CKAN's `status_show` and `help_show` actions.

Usage

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

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

Arguments

url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)
name	(character) CKAN action name to describe.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/")
status_show()
help_show("package_search")

## End(Not run)
```

ds_create

Add a new table to a datastore

Description

BEWARE: This function does not work yet.

Usage

```

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

```

Arguments

<code>resource_id</code>	(string) Resource id that stores the data.
<code>resource</code>	(dictionary) Resource dictionary for resource_create() . Use it instead of <code>resource_id</code> (optional)
<code>force</code>	(logical) To edit a read-only resource, set to TRUE. Default: FALSE
<code>aliases</code>	(character) Names for read only aliases of the resource. (optional)
<code>fields</code>	(list) Fields/columns and their extra metadata. (optional)
<code>records</code>	(list) The data, for example: [{"dob": "2005", "some_stuff": ["a", "b"]}](optional)
<code>primary_key</code>	(character) Fields that represent a unique key (optional)
<code>indexes</code>	(character) Indexes on table (optional)
<code>include_records</code>	(logical) If TRUE, CKAN 2.12+ returns the actual inserted records (including <code>_id</code> values and transformations) in the response. Default: FALSE. See https://github.com/ckan/ckan/pull/8684 . Note: bulk inserts with <code>include_records = TRUE</code> can hit a server-side error on CKAN 2.12.0; use a single record or ds_upsert() with <code>include_records = TRUE</code> instead when affected.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

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

## End(Not run)
```

ds_create_dataset	<i>Datastore - create a new resource on an existing dataset</i>
-------------------	---

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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Details

This function is deprecated and will be defunct in the next version of this package. Use [resource_create\(\)](#) instead. For example, replace `ds_create_dataset(package_id, name, path)` with `resource_create(package_id, name = name, upload = path)`.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

References

https://docs.ckan.org/en/2.12/api/index.html#ckan.logic.action.create.resource_create

Examples

```
## 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_pid = "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 CKAN datastore

Description

Datastore - search or get a dataset from CKAN datastore

Usage

```
ds_search(
  resource_id = NULL,
  filters = NULL,
  q = NULL,
  plain = NULL,
  language = NULL,
  fields = NULL,
  offset = NULL,
  limit = NULL,
  sort = NULL,
  include_next_page = FALSE,
  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, for example {"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, for example score desc, or descending, for example score asc. You can sort by two fields, for example score desc, price asc. You can sort by a function, for example sum(x_f, y_f) desc, which sorts by the sum of x_f and y_f in descending order. (optional)

include_next_page	(logical) If TRUE, CKAN 2.12+ returns a next_page value with filters for fast keyset pagination. Ignored unless records are sorted by the _id field (optional, default: FALSE). See https://docs.ckan.org/en/2.12/maintaining/datastore.html#search-pagination .
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Details

From the help for this method "The datastore_search action allows you to search data in a resource." If a DataStore resource belongs to a private CKAN resource, you can read it only with access to that resource. You must send the appropriate authorization.

CKAN 2.12+ filters accept advanced syntax: range operations (lt, lte, gt, gte, eq), lists mixing values and ranges, nested AND/OR via lists, and \$or groups (see <https://docs.ckan.org/en/2.12/maintaining/datastore.html#filters>). The filters argument passes through untouched, so both classic (`{ "key1": "a" }`) and advanced filters work. For large tables prefer keyset pagination (`include_next_page = TRUE` with `sort = "_id asc"`) over large offset values.

If you set `plain=FALSE`, you enable the entire PostgreSQL *full text search query language*. You can find a listing of all available resources at the alias *table_metadata* full text search query language: <https://www.postgresql.org/docs/9.1/static/datatype-textsearch.html#DATATYPE-TSQUERY>

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://data.gov.au/")
rid <- "eef6a84b-ad44-446f-9cf9-fb5d135e3123"

ds_search(resource_id = rid)
ds_search(resource_id = rid, as = "table")
ds_search(resource_id = rid, as = "json")

ds_search(resource_id = rid, limit = 1, as = "table")
ds_search(resource_id = rid, q = "S*")

# Return selected fields
ds_search(
  resource_id = rid,
  fields = c("name", "amount"),
```

```

    as = "table"
  )

  # Match more than one field. CKAN applies the conditions together.
  ds_search(
    resource_id = rid,
    filters = list(status = "active", category = "water"),
    fields = c("name", "status", "category"),
    as = "table"
  )

  ## End(Not run)

```

ds_search_sql

*Datastore - search or get a dataset from CKAN datastore***Description**

Datastore - search or get a dataset from CKAN datastore

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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://data.gov.au/")
sql <- 'SELECT * from "eef6a84b-ad44-446f-9cf9-fb5d135e3123" LIMIT 2'
ds_search_sql(sql, as = "table")
sql2 <- 'SELECT "Dog","TRI" from "eef6a84b-ad44-446f-9cf9-fb5d135e3123" LIMIT 2'
ds_search_sql(sql2, as = "table")

## End(Not run)
```

files

File management

Description

These helpers wrap CKAN 2.12 `file_*` endpoints. Files are first-class entities in CKAN 2.12 (see <https://docs.ckan.org/en/2.12/api/#ckan-logic-action-file> and <https://github.com/ckan/ckan/pull/9026>). The target CKAN instance must run CKAN 2.12 or later; each wrapper calls `ensure_action_available()` so it fails clearly on older instances.

Usage

```
file_create(
  upload,
  name = NULL,
  storage = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

file_register(
  location = NULL,
  storage = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

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

```
)

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

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

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

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

file_ownership_transfer(
    id,
    owner_id,
    owner_type,
    force = FALSE,
    pin = FALSE,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

file_owner_scan(
```

```

    owner_id,
    owner_type,
    start = NULL,
    rows = NULL,
    sort = NULL,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

```

Arguments

upload	(character) Local path of the file to upload for <code>file_create()</code> . Required.
name	(character) Human-readable file name. For <code>file_create()</code> , defaults to the uploaded file's name when upload is a local path.
storage	(character) Name of the storage handling the upload. Defaults to the configured default storage.
url	Base URL to use. Default: <code>https://demo.ckan.org/</code> . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)
location	(character) Location of the file in the storage for <code>file_register()</code> . Required.
id	(character or <code>ckan_file</code>) File identifier.
owner_id	(character) ID of the new owner for <code>file_ownership_transfer()</code> and <code>file_owner_scan()</code> .
owner_type	(character) Type of the new owner (for example "package", "user", "group", "organization").
force	(logical) For <code>file_ownership_transfer()</code> , move the file even if it is pinned. Default: FALSE.
pin	(logical) For <code>file_ownership_transfer()</code> , pin the file after transfer to stop future transfers. Default: FALSE.
start	(numeric) Index of the first row for <code>file_owner_scan()</code> .
rows	(numeric) Number of rows to return for <code>file_owner_scan()</code> .
sort	(character) File column used for sorting in <code>file_owner_scan()</code> . Optional; when NULL the server default ("name") applies.

Details

By default only sysadmins may call `file_create()`; see `ckan.files.authenticated_uploads.allow` in the CKAN docs for granting access to registered users.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
f <- file_create(upload = "path/to/file.csv")
file_show(f$id)

## End(Not run)
```

follow_objects

Follow and unfollow CKAN datasets or groups

Description

Create or remove follower relationships for a CKAN object. You must authenticate as a user with an API key. The key represents the follower.

Usage

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

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

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

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

```

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

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

```

Arguments

id	(character or ckan_*) Identifier for the object to inspect. For datasets, pass an id/slug or ckan_package. For groups, pass the corresponding identifier or ckan_group. For organizations, pass the corresponding identifier or ckan_organization. For users, pass a username or ckan_user.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the as parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
dataset_follow("my-dataset")
dataset_am_following("my-dataset")
dataset_unfollow("my-dataset")

## End(Not run)

```

follow_users	<i>Follow or unfollow users</i>
--------------	---------------------------------

Description

Manage user-to-user follower relationships.

Usage

```

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

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

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

```

Arguments

<code>id</code>	(character or <code>ckan_user</code>) The user to follow or inspect.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
follow_user("demo-user")
am_following_user("demo-user")
unfollow_user("demo-user")

## End(Not run)

```

followee_counts	<i>Followee counts for CKAN users</i>
-----------------	---------------------------------------

Description

Count how many objects of each type (or overall) a user follows.

See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.organization_followee_count for the official API contract.

Usage

```
followee_count(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
user_followee_count(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
dataset_followee_count(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
group_followee_count(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
organization_followee_count(  
    id,  
    url = get_default_url(),
```

```

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

```

Arguments

<code>id</code>	(character or <code>ckan_user</code>) User identifier.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
followee_count("demo-user")
user_followee_count("demo-user")
dataset_followee_count("demo-user")
organization_followee_count("demo-user")

## End(Not run)

```

<code>followee_lists</code>	<i>Followee lists for CKAN users</i>
-----------------------------	--------------------------------------

Description

The function returns the objects that a user follows.

See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.organization_followee_list for the official API contract.

Usage

```
followee_list(  
  id,  
  q = NULL,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)  
  
user_followee_list(  
  id,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)  
  
dataset_followee_list(  
  id,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)  
  
group_followee_list(  
  id,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)  
  
organization_followee_list(  
  id,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)
```

Arguments

id	(character or ckan_user) User identifier.
q	(character) Optional prefix filter for followee_list().
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
followee_list("demo-user", q = "ckan")
dataset_followee_list("demo-user")
organization_followee_list("demo-user")

## End(Not run)
```

follower_counts	<i>Follower counts for CKAN objects</i>
-----------------	---

Description

Report follower totals for datasets, groups, organizations, or users.

See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.organization_follower_count for the official API contract.

Usage

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

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

```

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

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

```

Arguments

<code>id</code>	(character or <code>ckan_*</code>) Identifier for the object to inspect. For datasets, pass an id/slug or <code>ckan_package</code> . For groups, pass the corresponding identifier or <code>ckan_group</code> . For organizations, pass the corresponding identifier or <code>ckan_organization</code> . For users, pass a username or <code>ckan_user</code> .
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
dataset_follower_count("my-dataset")
group_follower_count("my-group")
organization_follower_count("my-organization")
user_follower_count("demo-user")

## End(Not run)

```

follower_lists	<i>Follower lists for CKAN objects</i>
----------------	--

Description

The function returns the individual follower records for CKAN datasets, groups, organizations, or users.

See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.organization_follower_list for the official API contract.

Usage

```
dataset_follower_list(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
group_follower_list(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
user_follower_list(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
organization_follower_list(  
    id,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)
```

Arguments

id	(character or ckan_*) Identifier for the object to inspect. For datasets, pass an id/slug or ckan_package. For groups, pass the corresponding identifier or ckan_group. For organizations, pass the corresponding identifier or ckan_organization. For users, pass a username or ckan_user.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
dataset_follower_list("my-dataset")
group_follower_list("my-group")
organization_follower_list("my-organization")
user_follower_list("demo-user")

## End(Not run)
```

format_autocomplete	<i>Autocomplete resource formats.</i>
---------------------	---------------------------------------

Description

Return a list of resource formats whose names contain a query string. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.format_autocomplete for the official API contract.

Usage

```
format_autocomplete(
  q,
  limit = 5,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<code>q</code>	(character) Partial string to search for. The function returns entries whose name or title contains this string. (required)
<code>limit</code>	(numeric) The maximum number of resource formats to return. Default: 5 (optional)
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A character vector of matching format names. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.format_autocomplete

Examples

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

format_autocomplete(q = "csv")

## End(Not run)
```

get_site_user

Show the CKAN site user.

Description

Return the CKAN site user. The site user is a special internal user that the web interface and background jobs use for internal operations. You must be a sysadmin to call this endpoint. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.get_site_user for the official API contract.

Usage

```
get_site_user(
  defer_commit = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<code>defer_commit</code>	(logical) By default (FALSE), <code>get_site_user</code> commits and cleans up the current transaction. If TRUE, the caller is responsible for committing the transaction after the call. Leaving open connections can cause CLI commands to hang (optional).
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A `ckan_user` object. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.get_site_user

Examples

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

get_site_user()

## End(Not run)
```

group_autocomplete	<i>Autocomplete group names.</i>
--------------------	----------------------------------

Description

Return a list of group names that contain a query string. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.group_autocomplete for the official API contract.

Usage

```
group_autocomplete(  
  q,  
  limit = 20,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)
```

Arguments

q	(character) Partial string to search for. The function returns entries whose name or title contains this string. (required)
limit	(numeric) The maximum number of groups to return. Default: 20 (optional)
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list of group dictionaries, each with the keys name, title, and id. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.group_autocomplete

Examples

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

group_autocomplete(q = "data")

## End(Not run)
```

group_create	Create a group
--------------	----------------

Description

Create a group

Usage

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

Arguments

name	(character) the name of the new dataset. It must be between 2 and 100 characters long and contain only lowercase alphanumeric characters, • and <code>_</code>, for example <code>'warandpeace'</code>
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 of an image for the group's page (optional)

type	(character) The type of the dataset (optional). IDatasetForm plugins link to different dataset types. They provide custom behaviour for handling these types
state	(character) The current state of the dataset, for example 'active' or 'deleted' (optional, default: 'active'). Only active datasets appear in search results and other lists of datasets. If you lack permission to change the state of the dataset, the function ignores this parameter.
approval_status	(character) Approval status (optional)
extras	(list of dataset extra dictionaries) The dataset's extras (optional). Extras are arbitrary (key: value) metadata items for datasets. Each extra dictionary must have keys 'key' (a string) and 'value' (a string)
packages	(data.frame) The datasets (packages) that belong to the group. It is a data.frame. Each row has column 'name' (string, the id or name of the dataset) and optionally 'title' (string, the title of the dataset)
groups	(data.frame) The groups to which the dataset belongs (optional). Each data.frame row must have one or more of the following columns that 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. It is a list of dictionaries. Each dictionary has key 'name' (string, the id or name of the user) and optionally 'capacity' (string, the capacity in which the user is a member of the group)
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

## End(Not run)
```

group_delete	<i>Delete a group</i>
--------------	-----------------------

Description

Delete a group

Usage

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

Arguments

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

Value

(logical) TRUE on success

Examples

```
## 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)

## End(Not run)
```

group_list	<i>List groups.</i>
------------	---------------------

Description

List groups.

Usage

```
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 order, for example score desc. You can specify descending order, for example score asc. You can sort by two fields, for example score desc, price asc. You can sort by a function, for example sum(x_f, y_f) desc. The function then sorts by the sum of x_f and y_f in descending order.
groups	(character) A list of names of the groups to return. If you give this list, the function returns only groups with names in this list.
all_fields	(logical) Return full group dictionaries instead of names. Default: FALSE
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

## End(Not run)
```

group_list_authz	<i>List groups the current user can edit</i>
------------------	--

Description

List groups the current user can edit

Usage

```
group_list_authz(
  type = "group",
  groups = NULL,
  all_fields = FALSE,
  include_dataset_count = TRUE,
  include_extras = FALSE,
  include_tags = FALSE,
  include_groups = FALSE,
  include_users = FALSE,
  available_only = FALSE,
  am_member = FALSE,
  offset = NULL,
  limit = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

type	(character) Group type, defaults to "group".
groups	(character vector) Optional list of group names to filter.
all_fields	(logical) Return full group dictionaries.
include_dataset_count, include_extras, include_tags, include_groups, include_users	(logical) Include additional metadata when all_fields = TRUE.
available_only	(logical) Remove groups already associated to the context dataset.
am_member	(logical) Only return memberships for the current user.

offset	(numeric) Where to start getting activity items from (optional, default: 0)
limit	(numeric) The maximum number of activities to return (optional, default: 31)
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

## End(Not run)
```

group_member_create	<i>Make a user a member of a group</i>
---------------------	--

Description

Make a user a member of a group

Usage

```
group_member_create(
  id,
  username,
  role,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character, ckan_group, or ckan_organization) Identifier for the container.
username	(character or ckan_user) User to add.
role	(character) Desired role ("member", "editor", "admin").
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
group_member_create("my-group", username = "demo-user", role = "member")

## End(Not run)
```

group_member_delete	<i>Remove a user from a group</i>
---------------------	-----------------------------------

Description

Remove a user from a group

Usage

```
group_member_delete(
  id,
  username,
  url = get_default_url(),
  key = get_default_key(),
  ...
)
```

Arguments

id	(character, ckan_group, or ckan_organization) Identifier for the container.
username	(character or ckan_user) User to add.
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE on success

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
group_member_delete("my-group", username = "demo-user")

## End(Not run)
```

group_package_show	<i>Show the datasets of a group.</i>
--------------------	--------------------------------------

Description

Return the datasets (packages) that belong to a group. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.group_package_show for the official API contract.

Usage

```
group_package_show(
  id,
  limit = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character or ckan_group) The id or name of the group, or a ckan_group object.
limit	(numeric) The maximum number of datasets to return (optional).
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list of `ckan_package` objects. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.group_package_show

Examples

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

res <- group_list()
group_package_show(res[[1]]$name)
group_package_show(res[[1]], limit = 5, as = "table")

## End(Not run)
```

group_patch	<i>Update a group's metadata</i>
-------------	----------------------------------

Description

Update a group's metadata

Usage

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

Arguments

x	(list) A list with key-value pairs
id	(character) Resource ID to update (required)
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## 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_purge

Purge a group

Description

IMPORTANT: You must be a sysadmin to purge a group. Purging a group cannot be undone: it completely removes the group from the CKAN database, whereas deleting a group only marks it as deleted. Datasets in the group remain, just no longer in the purged group. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.delete.group_purge for the official API contract.

Usage

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

Arguments

<code>id</code>	(character or <code>ckan_group</code>) The name or id of the group to purge, or a <code>ckan_group</code> object.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.delete.group_purge

Examples

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

# create a group
(res <- group_create("foobar-group",
  title = "Foo bars",
  description = "love foo bars"
))

# delete the group just created
res$id
group_delete(id = res$id)

# purge the group just deleted
res$id
group_purge(id = res$id)

## End(Not run)
```

group_show	<i>Show a group</i>
------------	---------------------

Description

Show a group

Usage

```
group_show(
  id,
  include_datasets = TRUE,
  include_users = 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
include_users	(logical) Include the group's users. CKAN 2.12 changed the server default to FALSE regardless of the <code>ckan.auth.public_user_details</code> setting (https://github.com/ckan/ckan/pull/9232); <code>ckanr</code> defaults to TRUE to preserve the historical behavior. The parameter is accepted on all supported CKAN versions (2.9, 2.10, 2.11, 2.12).
url	Base URL to use. Default: <code>https://demo.ckan.org/</code> . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Details

By default the function drops the help and success slots. It returns only the result slot. If you want raw json, request `as = 'json'`. Then you parse the result yourself to get the help slot.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## 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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## 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)
```

job_queue

Background job helpers

Description

Manage the RQ/Redis job queue of CKAN. You must have sysadmin access.

Usage

```
job_list(
  queues = NULL,
  limit = NULL,
  ids_only = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

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

```

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

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

```

Arguments

queues	(character) Queue names to target.
limit	(numeric) Maximum number of jobs to return. CKAN 2.12 returns 200 jobs by default; use <code>limit</code> or the <code>ckan.jobs.default_list_limit</code> config option to change it (https://github.com/ckan/ckan/pull/8070).
ids_only	(logical) Return only job IDs. Default: FALSE.
url	Base URL to use. Default: <code>https://demo.ckan.org/</code> . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)
id	(character) Job identifier.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-ckan-key")
job_list()

## End(Not run)

```

license_list	<i>Return the list of licenses available for datasets on the site.</i>
--------------	--

Description

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

Usage

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

Arguments

id	(character) Package identifier.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

member_create	Create or update a membership via member_* endpoints
---------------	--

Description

Create or update a membership via member_* endpoints

Usage

```
member_create(
  id,
  object,
  object_type,
  capacity,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character, ckan_group, or ckan_organization) Identifier for the container.
object	(character) The object to add (dataset id, user name, etc.).
object_type	(character) Type of object you add ("user", "package", ...).
capacity	(character) Capacity for the membership.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the as parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
member_create("my-group", object = "demo-user", object_type = "user", capacity = "member")

## End(Not run)
```

member_delete	<i>Remove a membership via member_* endpoints</i>
---------------	---

Description

Remove a membership via member_* endpoints

Usage

```
member_delete(
  id,
  object,
  object_type,
  url = get_default_url(),
  key = get_default_key(),
  ...
)
```

Arguments

id	(character, ckan_group, or ckan_organization) Identifier for the container.
object	(character) The object to add (dataset id, user name, etc.).
object_type	(character) Type of object you add ("user", "package", ...).
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE on success

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
member_delete("my-group", object = "demo-user", object_type = "user")

## End(Not run)
```

member_list	<i>List members for a group or organization</i>
-------------	---

Description

List members for a group or organization

Usage

```
member_list(
  id,
  object_type = NULL,
  capacity = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character, ckan_group, or ckan_organization) Identifier for the container.
object_type	(character) Optional object type filter ("user", "package", etc.).
capacity	(character) Optional capacity filter ("member", "editor", "admin").
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
member_list("my-group", object_type = "user")

## End(Not run)
```

member_roles_list	<i>Report available member roles</i>
-------------------	--------------------------------------

Description

Report available member roles

Usage

```
member_roles_list(
  group_type = "organization",
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

group_type	(character) Either "group" or "organization".
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the as parameter.

Examples

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

## End(Not run)
```

membership	<i>Membership helpers</i>
------------	---------------------------

Description

Utilities help you manage CKAN group and organization membership endpoints.

organization_autocomplete

Autocomplete organization names.

Description

Return a list of organization names that contain a query string. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.organization_autocomplete for the official API contract.

Usage

```
organization_autocomplete(
  q,
  limit = 20,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<code>q</code>	(character) Partial string to search for. The function returns entries whose name or title contains this string. (required)
<code>limit</code>	(numeric) The maximum number of organizations to return. Default: 20 (optional)
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list of organization dictionaries, each with the keys name, title, and id. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.organization_autocomplete

Examples

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

organization_autocomplete(q = "data")

## End(Not run)
```

organization_create	<i>Create an organization</i>
---------------------	-------------------------------

Description

Create an organization

Usage

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

Arguments

name	(character) the name of the organization. It is a string between 2 and 100 characters long. It contains 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 of an image for the organization's page (optional)

state	(character) the current state of the organization, for example 'active' or 'deleted' (optional). Only active organizations appear in search results and other lists of organizations. If you lack permission to change the state of the organization, the function ignores this parameter. Default: 'active'
approval_status	(character) Approval status
extras	The organization's extras (optional). Extras are arbitrary (key: value) metadata items for organizations. Each extra dictionary must have keys 'key' (a string) and 'value' (a string). See <code>package_relationship_create</code> for the format of relationship dictionaries (optional)
packages	(list of dictionaries) the datasets (packages) that belong to the organization. It is a list of dictionaries. Each dictionary has 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. It is a list of dictionaries. Each dictionary has 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: <code>https://demo.ckan.org/</code> . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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	<i>Delete an organization</i>
---------------------	-------------------------------

Description

Delete an organization

Usage

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

Arguments

id	(character) name or id of the organization
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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	<i>List organizations</i>
-------------------	---------------------------

Description

List organizations

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). It shows if the sort order decreases.
organizations	(character or NULL). A list of names of the organizations to return. NULL returns all organizations.
all_fields	(logical). Return the name or all fields of the object.
limit	(numeric) The maximum number of organizations to return (optional, default: 31)
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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_list_for_user

List organizations a user can act on

Description

List organizations a user can act on

Usage

```
organization_list_for_user(
  id = NULL,
  permission = "manage_group",
  include_dataset_count = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character or ckan_user) Optional user identifier. It defaults to the authenticated user.
permission	(character) Permission to check (defaults to "manage_group").
include_dataset_count	(logical) Include dataset counts in the response.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to `data.frame`'s when possible. The result can vary from a vector, list or `data.frame`. (required)

... Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

## End(Not run)
```

organization_member_create

Make a user a member of an organization

Description

Make a user a member of an organization

Usage

```
organization_member_create(
  id,
  username,
  role,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id (character, `ckan_group`, or `ckan_organization`) Identifier for the container.

username (character or `ckan_user`) User to add.

role (character) Desired role ("member", "editor", "admin").

url Base URL to use. Default: `https://demo.ckan.org/`. 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 the table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to `data.frame`'s when possible. The result can vary from a vector, list or `data.frame`. (required)

... Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
organization_member_create("my-organization", username = "demo-user", role = "member")

## End(Not run)
```

organization_member_delete

Remove a user from an organization

Description

Remove a user from an organization

Usage

```
organization_member_delete(
  id,
  username,
  url = get_default_url(),
  key = get_default_key(),
  ...
)
```

Arguments

id (character, `ckan_group`, or `ckan_organization`) Identifier for the container.

username (character or `ckan_user`) User to add.

url Base URL to use. Default: `https://demo.ckan.org/`. See also [ckanr_setup](#) and [get_default_url](#).

key A privileged CKAN API key. Default: your key set with [ckanr_setup](#)

... Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

(logical) TRUE on success

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
organization_member_delete("my-organization", username = "demo-user")

## End(Not run)
```

organization_patch	<i>Patch an organization.</i>
--------------------	-------------------------------

Description

This function partially updates an organization: it updates only the provided parameters and leaves all other parameters unchanged (unlike [organization_update\(\)](#), which may delete parameters not explicitly provided). See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.patch.organization_patch for the official API contract.

Usage

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

Arguments

x	(list) A list with key-value pairs
id	(character or ckan_organization) The id or name of the organization to patch, or a ckan_organization object.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

The patched organization as a `ckan_organization` object. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.patch.organization_patch

Examples

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

# Create an organization
(res <- organization_create("hello-my-org2"))

# Get the organization
org <- organization_show(res$id)

# Make some changes
x <- list(title = "!hello world!", description = "hello world org")
organization_patch(x, id = org)

## End(Not run)
```

organization_purge	<i>Purge an organization</i>
--------------------	------------------------------

Description

IMPORTANT: You must be a sysadmin to purge an organization. Once you purge an organization, the system removes it permanently.

Usage

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

Arguments

id	(character) name or id of the organization
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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)

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

## End(Not run)
```

organization_show	<i>Show an organization</i>
-------------------	-----------------------------

Description

Show an organization

Usage

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

Arguments

<code>id</code>	(character) Organization id or name.
<code>include_datasets</code>	(logical). Include a list of the organization datasets.
<code>include_users</code>	(logical) Include the organization's users. CKAN 2.12 changed the server default to FALSE regardless of the <code>ckan.auth.public_user_details</code> setting (https://github.com/ckan/ckan/pull/9232); ckanr defaults to TRUE to preserve the historical behavior. The parameter is accepted on all supported CKAN versions (2.9, 2.10, 2.11, 2.12).
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Details

By default the function drops the help and success slots. It returns only the result slot. If you want raw json, request `as = 'json'`. Then you parse the result yourself to get the help slot.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## 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)
```

organization_update *Update an organization.*

Description

This function updates all organization metadata fields. You must be authorized to edit the organization. Update methods may delete parameters not explicitly provided: if you want to edit only specific attributes, use [organization_patch\(\)](#) instead. For the full list of accepted fields, see [organization_create\(\)](#) and https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.organization_update.

Usage

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

Arguments

x	(list) A list with key-value pairs
id	(character or ckan_organization) The name or id of the organization to update, or a ckan_organization object.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

The updated organization as a ckan_organization object. With `as = "table"`, a data.frame; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.organization_update

Examples

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

# First, create an organization
org <- organization_create("water-bears2")
organization_show(org)

# Make some changes
x <- list(description = "An organization about water bears")

# Then update the organization
organization_update(x, id = org)

## 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

<code>id</code>	(character) Package identifier.
<code>offset</code>	(numeric) Where to start getting activity items from (optional, default: 0)
<code>limit</code>	(numeric) The maximum number of activities to return (optional, default: 31)
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup

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

... Extra curl arguments. The function passes them to `verb-POST` (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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$id, as = "table")
package_activity_list(res$id, as = "json")

## End(Not run)
```

`package_autocomplete` *Autocomplete dataset names.*

Description

Return a list of datasets (packages) with names or titles that contain a query string. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.package_autocomplete for the official API contract.

Usage

```
package_autocomplete(
  q,
  limit = 10,
```

```

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

```

Arguments

<code>q</code>	(character) Partial string to search for. The function returns entries whose name or title contains this string. (required)
<code>limit</code>	(numeric) The maximum number of datasets to return. Default: 10 (optional)
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list of dataset dictionaries. Each dictionary has at least the keys name and title. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.package_autocomplete

Examples

```

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

package_autocomplete(q = "spending")
package_autocomplete(q = "spending", limit = 5, as = "table")

## End(Not run)

```

package_collaborator_create

Create or update a dataset collaborator

Description

Create or update a dataset collaborator

Usage

```
package_collaborator_create(
  id,
  user_id,
  capacity,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<code>id</code>	(character or <code>ckan_package</code>) Dataset identifier or object.
<code>user_id</code>	(character or <code>ckan_user</code>) User identifier or object.
<code>capacity</code>	(character) Collaborator role ("member", "editor", "admin").
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
package_collaborator_create("my-dataset", "new-user", capacity = "editor")

## End(Not run)
```

```
package_collaborator_delete
```

Delete a dataset collaborator

Description

Delete a dataset collaborator

Usage

```
package_collaborator_delete(
  id,
  user_id,
  url = get_default_url(),
  key = get_default_key(),
  ...
)
```

Arguments

id	(character or ckan_package) Dataset identifier or object.
user_id	(character or ckan_user) User identifier or object.
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE when the function deletes the collaborator successfully.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
package_collaborator_delete("my-dataset", "new-user")

## End(Not run)
```

package_collaborator_list

List collaborators for a dataset

Description

List collaborators for a dataset

Usage

```
package_collaborator_list(
  id,
  capacity = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<code>id</code>	(character or <code>ckan_package</code>) Dataset identifier or object.
<code>capacity</code>	(character) Optional capacity filter ("member", "editor", "admin").
<code>url</code>	Base URL to use. Default: <code>https://demo.ckan.org/</code> . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
package_collaborator_list("my-dataset")

## End(Not run)
```

`package_collaborator_list_for_user`

List packages a user collaborates on

Description

List packages a user collaborates on

Usage

```
package_collaborator_list_for_user(
  id,
  capacity = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character or ckan_user) User identifier or object.
capacity	(character) Optional capacity filter.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
package_collaborator_list_for_user("sckottie")

## End(Not run)
```

package_collaborators *Package collaborator utilities*

Description

Package collaborator utilities

package_create *Create a package*

Description

Create a package

Usage

```

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

<code>name</code>	(character) the name of the new dataset. It must be between 2 and 100 characters long and contain only lowercase alphanumeric characters, - and _, for example 'warandpeace'
<code>title</code>	(character) the title of the dataset (optional, default: same as name)
<code>private</code>	(logical) whether the dataset is private (optional, default: FALSE). If TRUE, you must give a value for <code>owner_org</code> .
<code>author</code>	(character) the name of the dataset's author (optional)
<code>author_email</code>	(character) the email address of the dataset's author (optional)
<code>maintainer</code>	(character) the name of the dataset's maintainer (optional)
<code>maintainer_email</code>	(character) the email address of the dataset's maintainer (optional)
<code>license_id</code>	(license id string) The id of the dataset's license. See <code>license_list()</code> for available values (optional)
<code>notes</code>	(character) a description of the dataset (optional)
<code>package_url</code>	(character) a URL for the dataset's source (optional)

version	(string, no longer than 100 characters) (optional)
state	(character) the current state of the dataset, for example 'active' or 'deleted' (optional, default: 'active'). Only active datasets appear in search results and other lists of datasets. If you lack permission to change the state of the dataset, the function ignores this parameter.
type	(character) the type of the dataset (optional). IDatasetForm plugins link to different dataset types. They provide custom behaviour for handling these types
resources	(list of resource dictionaries) The dataset's resources. See resource_create() for the format of resource dictionaries (optional)
tags	(list of tag dictionaries) The dataset's tags. See tag_create() for the format of tag dictionaries (optional)
extras	(list of dataset extra dictionaries) The dataset's extras (optional). Extras are arbitrary (key: value) metadata items for datasets. Each extra dictionary must have keys 'key' (a string) and '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	(data.frame) the groups to which the dataset belongs. Each row must have one or more of the following columns that 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() . See example (optional)
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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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"))

# include a group
package_create("brownbear", groups = data.frame(id = "some-id"))

## End(Not run)
```

```
package_create_default_resource_views
```

```
Create default views for all resources in a dataset
```

Description

Create default views for all resources in a dataset

Usage

```
package_create_default_resource_views(
  package,
  create_datastore_views = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

package	(character or ckan_package) Dataset identifier.
create_datastore_views	(logical) When TRUE, the function creates only views that require DataStore-backed resources. The function uses this option when DataPusher finishes ingesting data.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

## End(Not run)
```

package_dataset_extras

Dataset maintenance helpers

Description

The file covers extra CKAN 2.11 dataset endpoints for bulk edits and destructive actions.

package_delete

Delete a package

Description

Delete a package

Usage

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

Arguments

<code>id</code>	(character) The id of the package. Required.
<code>url</code>	Base url to use. Default: <code>https://data.ontario.ca</code> See also ckanr_setup() and get_default_url()
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>...</code>	Curl args. The function sends them to crul::verb-POST (optional)

Value

(logical) TRUE when the function deletes the package successfully.

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	<i>List datasets.</i>
--------------	-----------------------

Description

List datasets.

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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
package_list()
package_list(as = "json")
package_list(as = "table")

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

## End(Not run)
```

`package_list_current` *List current packages with resources.*

Description

List current packages with resources.

Usage

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

Arguments

<code>offset</code>	(numeric) Where to start getting activity items from (optional, default: 0)
<code>limit</code>	(numeric) The maximum number of activities to return (optional, default: 31)
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

## End(Not run)
```

package_owner_org_update

Move a dataset to another organization

Description

On CKAN 2.12 `package_owner_org_update` may report success without moving the dataset; when the verification shows the dataset is still owned by the previous organization, the function falls back to `package_patch()` with `owner_org`, which moves the dataset reliably on all supported versions.

Usage

```
package_owner_org_update(
  id,
  organization_id,
  url = get_default_url(),
  key = get_default_key(),
  ...
)
```

Arguments

<code>id</code>	(character or <code>ckan_package</code>) Dataset identifier.
<code>organization_id</code>	(character or <code>ckan_organization</code>) Owning organization identifier.
<code>url</code>	Base URL to use. Default: <code>https://demo.ckan.org/</code> . See also <code>ckanr_setup</code> and <code>get_default_url</code> .
<code>key</code>	A privileged CKAN API key. Default: your key set with <code>ckanr_setup</code>
<code>...</code>	Extra curl arguments. The function passes them to <code>verb-POST</code> (optional)

Value

(logical) TRUE when CKAN moves the dataset to the new organization successfully.

Examples

```
## Not run:
package_owner_org_update("dataset-id", organization_id = "target-org")

## End(Not run)
```

package_patch	<i>Update a package's metadata</i>
---------------	------------------------------------

Description

If the portal uses the `ckanext-scheming` extension for custom schema fields, put each custom field directly in `x`. Do not pass custom fields through `extras`. CKAN rejects custom fields in `extras` with an error.

Usage

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

Arguments

<code>x</code>	(list) A list with key-value pairs. Put <code>ckanext-scheming</code> custom fields here as named items.
<code>id</code>	(character) Resource ID to update (optional). If <code>x</code> lacks an "id" field, this parameter is required.
<code>extras</code>	(list) The dataset's extras (optional). Extras are arbitrary (key: value) metadata items for datasets. Each extra dictionary must have keys 'key' (a string) and 'value' (a string). Use this only for plain CKAN extras, not for <code>ckanext-scheming</code> custom fields.
<code>http_method</code>	(character) Which HTTP method (verb) to use. Use one of "GET" or "POST". Default: "GET"
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .

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

... Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

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

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

# Update a ckanext-scheming custom field: put the field in `x`
package_patch(list(id = res$id, internal_notes = "New internal notes"))

## End(Not run)
```

package_relationship_create

Create a dataset relationship

Description

Create a dataset relationship

Usage

```
package_relationship_create(
  subject,
  object,
  relationship_type,
  comment = NULL,
  url = get_default_url(),
  key = get_default_key(),
```

```

    as = "list",
    ...
  )

```

Arguments

subject	(character or ckan_package) Dataset that acts as the subject in the relationship.
object	(character or ckan_package) Dataset that acts as the object in the relationship.
relationship_type	(character) Relationship type ("depends_on", "derives_from", etc.).
comment	(character) Optional note for the relationship.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
package_relationship_create("dataset-a", "dataset-b", relationship_type = "derives_from")

## End(Not run)

```

```
package_relationship_delete
```

Delete a dataset relationship

Description

Delete a dataset relationship

Usage

```
package_relationship_delete(
  subject,
  object,
  relationship_type,
  url = get_default_url(),
  key = get_default_key(),
  ...
)
```

Arguments

subject	(character or ckan_package) Dataset that acts as the subject in the relationship.
object	(character or ckan_package) Dataset that acts as the object in the relationship.
relationship_type	(character) Relationship type ("depends_on", "derives_from", etc.).
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE when the function deletes the relationship successfully.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
package_relationship_delete("dataset-a", "dataset-b", relationship_type = "derives_from")

## End(Not run)
```

package_relationship_update

Update an existing dataset relationship

Description

Update an existing dataset relationship

Usage

```
package_relationship_update(
  subject,
  object,
  relationship_type,
  comment = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

subject	(character or ckan_package) Dataset that acts as the subject in the relationship.
object	(character or ckan_package) Dataset that acts as the object in the relationship.
relationship_type	(character) Relationship type ("depends_on", "derives_from", etc.).
comment	(character) Optional note for the relationship.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
package_relationship_update("dataset-a", "dataset-b", relationship_type = "derives_from")

## End(Not run)
```

package_relationships *Package relationship helpers*

Description

Utilities help you manage dataset relationships through CKAN's `package_relationship_*` endpoints.

package_relationships_list

List relationships for a dataset

Description

List relationships for a dataset

Usage

```
package_relationships_list(
  id,
  id2 = NULL,
  relationship_type = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character or ckan_package) The dataset to inspect.
id2	(character or ckan_package) Optional secondary dataset identifier to scope results.
relationship_type	(character) Filter on relationship type (see package_relationship_create()).
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-key")
package_relationships_list("my-dataset")

## End(Not run)
```

`package_resource_reorder`*Reorder resources for a dataset*

Description

Reorder resources for a dataset

Usage

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

Arguments

<code>id</code>	(character or <code>ckan_package</code>) Dataset identifier.
<code>order</code>	(character vector) Resource IDs in the desired order.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:  
pkg <- package_show("my-dataset")  
package_resource_reorder(pkg,  
  order = rev(vapply(pkg$resources, `[`, "id", FUN.VALUE = character(1)))  
)  
  
## End(Not run)
```

package_revise	<i>Revise a dataset using match/filter/update semantics</i>
----------------	---

Description

Revise a dataset using match/filter/update semantics

Usage

```
package_revise(
  match = NULL,
  filter = NULL,
  update = NULL,
  include = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

match	(list) Key/value pairs that identify the dataset to revise. Unless you use flattened keys, this parameter is required.
filter	(character or list) Patterns that describe fields to remove before the update runs.
update	(list) Values to set after filtering. The values support flattened keys.
include	(character or list) Optional patterns that delimit which fields the response returns.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
package_revise(
  match = list(name = "source-dataset"),
  update = list(notes = "New description")
)

## End(Not run)
```

`package_revision_list` *Return a dataset (package's) revisions as a list of dictionaries.*

Description

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

Usage

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

Arguments

<code>id</code>	(character) Package identifier.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

# list package revisions
package_revision_list(res$id)

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

# list package revisions
package_revision_list(res$id)

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

## End(Not run)
```

package_search

Search for packages.

Description

Search for packages.

Usage

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

```

    as = "list",
    ...
)

```

Arguments

<code>q</code>	Query terms, defaults to <code>':'</code> , or everything.
<code>fq</code>	Filter query. It does not affect the search. It only controls what the function returns.
<code>sort</code>	Field to sort on. You can specify ascending order, for example <code>score desc</code> . You can specify descending order, for example <code>score asc</code> . You can sort by two fields, for example <code>score desc, price asc</code> . You can sort by a function, for example <code>sum(x_f, y_f) desc</code> . The function then sorts by the sum of <code>x_f</code> and <code>y_f</code> in descending order.
<code>rows</code>	Number of records to return. Defaults to 10.
<code>start</code>	Record to start with. It defaults to the beginning.
<code>facet</code>	(logical) Whether to return facet results or not. Default: <code>FALSE</code>
<code>facet.limit</code>	(numeric) The maximum number of constraint counts for the facet fields. A negative value means unlimited. Default: 100. You can set it for each field.
<code>facet.field</code>	(character) This parameter sets a field to treat as a facet. It iterates over each Term in the field. It generates a facet count with that Term as the constraint. You can give this parameter multiple times for multiple facet fields. If you omit all field names for this parameter, the other parameters in this section have no effect.
<code>facet.mincount</code>	(integer) the minimum counts for facet fields. The results include only fields that meet this count.
<code>include_drafts</code>	(logical) If <code>TRUE</code> , the function includes draft datasets. A user gets only their own draft datasets. A sysadmin gets all draft datasets. Default: <code>FALSE</code> . First CKAN version: 2.6.1. If the CKAN version is older, or if the version is not available through <code>ckan_version()</code> , the function drops it from the request.
<code>include_private</code>	(logical) If <code>TRUE</code> , the function includes private datasets. It returns only private datasets from the user organizations. Sysadmins get all private datasets. Default: <code>FALSE</code> . First CKAN version: 2.6.1. If the CKAN version is older, or if the version is not available through <code>ckan_version()</code> , the function drops it from the request.
<code>use_default_schema</code>	(logical) Use default package schema instead of a custom schema from an IDataset-Form plugin. Default: <code>FALSE</code> . First CKAN version: 2.3.5. If the CKAN version is older, or if the version is not available through <code>ckan_version()</code> , the function drops it from the request.
<code>url</code>	Base URL to use. Default: <code>https://demo.ckan.org/</code> . See also <code>ckanr_setup</code> and <code>get_default_url</code> .
<code>key</code>	A privileged CKAN API key. Default: your key set with <code>ckanr_setup</code>

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

... Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## 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,
  http_method = "GET",
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<code>id</code>	(character/ckan_package) Package identifier, or a ckan_package object
<code>use_default_schema</code>	(logical) Use default package schema instead of a custom schema defined with an IDatasetForm plugin. Default: FALSE
<code>http_method</code>	(character) Which HTTP method (verb) to use. Use one of "GET" or "POST". Default: "GET"
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Details

By default the function drops the help and success slots. It returns only the result slot. If you want raw json, request as = 'json'. Then you parse the result yourself to get the help slot.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the as parameter.

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	<i>Update a package</i>
----------------	-------------------------

Description

This function updates all package metadata fields. Each update sets the metadata key "last_updated".

On CKAN < 2.12 the function overwrites any omitted metadata fields: any metadata fields missing from `x` are deleted in the package. On CKAN 2.12 and later the internal `allow_partial_update` context was removed (<https://github.com/ckan/ckan/pull/8155>): calling `package_update()` without resources now keeps existing resources instead of wiping them. To partially update nested fields on any version, prefer `package_patch()` or `package_revise()`.

CKAN 2.12+ also reports whether the update made a real change via a `changed_entities` envelope value (<https://github.com/ckan/ckan/pull/8407>); the value is part of the returned payload when the server provides it.

Usage

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

Arguments

<code>x</code>	(list) A list with key-value pairs
<code>id</code>	(character) Package identifier
<code>http_method</code>	(character) Which HTTP method (verb) to use. Use one of "GET" or "POST". Default: "GET"
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb=POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

# Step 1: get the dataset details as R list
ds_id <- "my-dataset-id-md5-hash"
ds <- ckanr::package_show(ds_id, as = "table")

# Step 2: update selected fields
ds$title <- "An updated title"
ds$description <- "Only title and description have been updated."
# ds contains all other package data, including tags and resources

# Step 3a: Update the dataset on CKAN with locally modified metadata `ds`
result <- ckanr::package_update(ds, ds_id)
# Replace existing package metadata

# Step 3b: Possible or intended data loss on CKAN < 2.12
# Any metadata fields missing from `ds` will be deleted in the package
# (on CKAN 2.12+ omitted `resources` are kept; use package_patch() or
# package_revise() for partial updates on any version)
del(ds$description)
result_with_deleted_description <- ckanr::package_update(ds, ds_id)

## End(Not run)
```

ping

Ping a CKAN server to test if it is up or down.

Description

Ping a CKAN server to test if it is up or down.

Usage

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

Arguments

<code>url</code>	Base URL to use. Default: <code>https://demo.ckan.org/</code> . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup

`as` (character) One of "logical" (default) or "json". With `as = "logical"` failures return FALSE. With `as = "json"` failures signal an error instead of returning a non-JSON logical.

`...` Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

(logical) TRUE when the CKAN server is up, FALSE when it is down. With `as = "json"`, the raw JSON response string instead.

Examples

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

## End(Not run)
```

<code>related_create</code>	<i>Create a related item</i>
-----------------------------	------------------------------

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 the package. You add the related item to this package. The value must be an alphanumeric string. Required.

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

`type` (character) Type of the related item. The value is one of API, application, idea, news article, paper, post or visualization. Required.

description	(character) Description of the related item. Optional.
related_id	(character) ID to assign to the related item. If the value is blank, the function assigns an ID. Optional.
related_url	(character) URL for the related item. Optional.
image_url	(character) URL of an image for the related item. Optional.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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 = "asfdaf", type = "idea")

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

## End(Not run)
```

related_delete	<i>Delete a related item.</i>
----------------	-------------------------------

Description

Delete a related item.

Usage

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

Arguments

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

Value

(logical) TRUE when the function deletes the related item successfully.

Examples

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

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

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

## End(Not run)
```

<code>related_list</code>	<i>List related items</i>
---------------------------	---------------------------

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: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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",
  ...
)
```

Arguments

<code>id</code>	(character) Related item identifier.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Details

By default the function drops the help and success slots, and it returns only the result slot. If you want raw json, pass `as = 'json'`. Then you parse the result to get the help slot.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

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

# get data back in different formats
related_show(res, as = "json")
```

```
related_show(res, as = "table")

## End(Not run)
```

resource_create	<i>Create a resource</i>
-----------------	--------------------------

Description

Create a resource

Usage

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

Arguments

package_id	(character) ID of the package. You add the resource to this package. The value must be an alphanumeric string. Required.
rcurl	(character) URL of the resource. Required.
revision_id	(character) Revision ID. Optional.
description	(character) Description of the resource. Optional. Required.

format	(character) Format. Optional.
hash	(character) Hash. Optional.
name	(character) Name of the resource. Optional. Required.
resource_type	(character) Resource type. Optional.
mimetype	(character) MIME type. Optional.
mimetype_inner	(character) Inner MIME type. 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) Path to a local file. Optional.
extras	(list) Extra metadata fields of the resource. Optional.
http_method	(character) HTTP method (verb) to use. The value is one of "GET" or "POST". Default is "GET".
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

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

package_create("foobbbbbbarrrr") |>
  resource_create(
    description = "my resource",
    name = "bearsareus",
    upload = file,
    extras = list(my_extra = "some value"),
    rcurl = "http://google.com"
  )

## End(Not run)
```

resource_create_default_resource_views

Create default views for a single resource

Description

Create default views for a single resource

Usage

```
resource_create_default_resource_views(
  resource,
  package = NULL,
  create_datastore_views = FALSE,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

resource	(character or ckan_resource) Resource identifier or object.
package	(optional) Dataset identifier or ckan_package object.

create_datastore_views	(logical) When TRUE, the function creates only views that require DataStore-backed resources. The function uses this option when DataPusher finishes ingesting data.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

## End(Not run)
```

resource_delete	<i>Delete a resource.</i>
-----------------	---------------------------

Description

Delete a resource.

Usage

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

Arguments

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

Value

(logical) TRUE when the function deletes the resource successfully.

Examples

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

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

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

# delete the resource
resource_delete(xx)

## End(Not run)
```

resource_patch	<i>Update a resource's metadata</i>
----------------	-------------------------------------

Description

To point an existing resource at a new external URL, pass the link through `rcurl`. If `x` also holds a `url` item, `rcurl` wins.

Usage

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

Arguments

<code>x</code>	(list) A list with key-value pairs
<code>id</code>	(character) Resource ID to update (required)

<code>rcurl</code>	(character) New external URL of the resource, for example an ArcGIS link. The function sends it as the resource url and sets <code>url_type</code> to <code>link</code> , so CKAN stores a true external link. If <code>x</code> holds its own <code>url_type</code> item, that value wins. Optional.
<code>url</code>	Base URL to use. Default: <code>https://demo.ckan.org/</code> . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of <code>list</code> (default), <code>table</code> , or <code>json</code> . Parsing with the <code>table</code> option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

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

# Get a resource
res <- resource_show(xx$id)
res$description

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

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

# Point the resource at a new external URL
zzz <- resource_patch(list(), id = res, rcurl = "https://example.com/data.geojson")
zzz$url

## End(Not run)
```

resource_search	<i>Search for resources.</i>
-----------------	------------------------------

Description

Search for resources.

Usage

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

Arguments

<code>q</code>	Query terms. The value is a string of the form <code>field:term</code> or a vector or list of strings. Each string uses the same form. <code>field</code> is a field or extra field on the Resource domain object. If <code>field</code> is <code>hash</code> , the function matches the term as a prefix of the <code>Resource.hash</code> field. If <code>field</code> is an extra field, the function matches against the extra fields stored against the Resource.
<code>sort</code>	Field to sort on. You can specify ascending (for example, <code>score desc</code>) or descending (for example, <code>score asc</code>). You can sort by two fields (for example, <code>score desc, price asc</code>). You can sort by a function (for example, <code>sum(x_f, y_f) desc</code>). The function sorts by the sum of <code>x_f</code> and <code>y_f</code> in descending order.
<code>offset</code>	Record to start at. The default is the beginning.
<code>limit</code>	Number of records to return.
<code>url</code>	Base URL to use. Default: <code>https://demo.ckan.org/</code> . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of <code>list</code> (default), <code>table</code> , or <code>json</code> . Parsing with the <code>table</code> option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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")
resource_search(q = c("description:encoded", "name:No.2"), url = "demo.ckan.org")

## End(Not run)
```

resource_show	<i>Show a resource.</i>
---------------	-------------------------

Description

Show a resource.

Usage

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

Arguments

- id (character) Resource identifier.
- url Base URL to use. Default: <https://demo.ckan.org/>. 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 the table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to `data.frame`'s when possible. The result can vary from a vector, list or `data.frame`. (required)
- ... Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

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

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

# show the resource
resource_show(xx$id)

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

## End(Not run)
```

resource_update

Update a resource

Description

You call this function to update a resource file attachment and "extra" metadata fields. Each update sets the metadata key "last_updated".

On CKAN < 2.12 omitted metadata fields could be overwritten; on CKAN 2.12+ unchanged resources are no longer revalidated and `package_update()` without resources keeps existing resources (<https://github.com/ckan/ckan/pull/8155>, <https://github.com/ckan/ckan/pull/5713>). To update selected metadata fields and keep all other fields unchanged, first retrieve the full resource metadata with `resource_show`. Then update this metadata as required. Then update the resource with `resource_update` with the locally updated metadata.

If you update a resource file, the new file must exist on a local path. You cannot use R objects directly to update a resource file. Instead, write them to a file. For example, use `tempfile()`. See the example.

To point an existing resource at a new external URL, pass the link through `rcurl`. Do not put the link in `extras`.

The CKAN base URL and API key default to the global options. `ckanr_setup` sets the global options.

Usage

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

Arguments

<code>id</code>	(character) Resource ID to update. Required.
<code>path</code>	(character) Local path of the file to upload. Optional.
<code>extras</code>	(list) Extra metadata fields of the resource. Optional.
<code>rcurl</code>	(character) New external URL of the resource, for example an ArcGIS link. The function sends it as the resource <code>url</code> and sets <code>url_type</code> to <code>link</code> , so CKAN stores a true external link. Use this without <code>path</code> to point an existing resource at a new external URL. If you also pass <code>path</code> , the uploaded file wins on the server. Optional.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of <code>list</code> (default), <code>table</code> , or <code>json</code> . Parsing with the <code>table</code> option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

The function returns the HTTP response from CKAN as a list (default), table, or JSON.

References

https://docs.ckan.org/en/2.12/api/index.html#ckan.logic.action.create.resource_create

Examples

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

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

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

# Modify dataset, here lowercase strings in one column
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)

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

# Update a resource's extra tags
## add extra tags without uploading a new file
resource_update(id,
  extras = list(some = "metadata")
)

# Point an existing resource at a new external URL
resource_update(xx$id, rcurl = "https://example.com/data.geojson")

## or remove all extra tags
resource_update(id, extras = list())

#####
# Using default settings
ckanr_setup(url = "https://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 = "https://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)

```

resource_view_clear	<i>Clear resource views</i>
---------------------	-----------------------------

Description

Clear resource views

Usage

```
resource_view_clear(  
  view_types = NULL,  
  url = get_default_url(),  
  key = get_default_key(),  
  ...  
)
```

Arguments

view_types	(character vector) Optional subset of view types to delete. When NULL, the function removes all views.
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE when the function clears the resource views successfully.

Examples

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

resource_view_create *Create a resource view*

Description

Create a resource view

Usage

```
resource_view_create(
  resource,
  view_type,
  title,
  description = NULL,
  config = NULL,
  filter_fields = NULL,
  filter_values = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

resource	(character or ckan_resource) Parent resource identifier.
view_type	(character) Plugin type registered on the CKAN site.
title	(character) Title of the view.
description	(character) Description. Optional.
config	(list) Arbitrary configuration list. The function passes it to the plugin.
filter_fields, filter_values	Optional filter parameters for filterable views.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
resource_view_create("resource-id", view_type = "image_view", title = "My view")

## End(Not run)
```

resource_view_delete *Delete a resource view*

Description

Delete a resource view

Usage

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

Arguments

id	(character or ckan_resource_view) View identifier.
url	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
key	A privileged CKAN API key. Default: your key set with ckanr_setup
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(bool) Result of the deletion. The value is TRUE when the function deletes the resource view successfully, and FALSE when it does not.

Examples

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

## End(Not run)
```

resource_view_list	<i>List resource views for a resource</i>
--------------------	---

Description

List resource views for a resource

Usage

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

Arguments

id	(character or ckan_resource) Resource identifier.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:  
pkg <- package_show("sample-dataset")  
resource_view_list(pkg$resources[[1]]$id)  
  
## End(Not run)
```

resource_view_reorder *Reorder resource views*

Description

Reorder resource views

Usage

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

Arguments

id	(character or ckan_resource) Resource identifier.
order	(character vector) View IDs in the desired order.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
resource_view_reorder("resource-id", order = c("view-id-1", "view-id-2"))

## End(Not run)
```

resource_view_show	<i>Show a resource view</i>
--------------------	-----------------------------

Description

Show a resource view

Usage

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

Arguments

id	(character or ckan_resource_view) View identifier.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

## End(Not run)
```

resource_view_update *Update an existing resource view*

Description

Update an existing resource view

Usage

```
resource_view_update(
  id,
  resource = NULL,
  title = NULL,
  description = NULL,
  config = NULL,
  filter_fields = NULL,
  filter_values = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

id	(character or ckan_resource_view) View identifier.
resource	(character or ckan_resource) Parent resource identifier.
title	(character) Title of the view.
description	(character) Description. Optional.
config	(list) Arbitrary configuration list. The function passes it to the plugin.
filter_fields, filter_values	Optional filter parameters for filterable views.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
resource_view_update("view-id", title = "Updated title")

## End(Not run)
```

resource_views	<i>Resource view management</i>
----------------	---------------------------------

Description

These helpers wrap CKAN resource_view_* endpoints.

revision_list	<i>Return a list of the IDs of the site's revisions.</i>
---------------	--

Description

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

Usage

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

Arguments

url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the as parameter.

Examples

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

## End(Not run)
```

servers

CKAN server URLs and other info

Description

CKAN server URLs and other info

Usage

```
servers()
```

Details

CKAN instances from <https://github.com/ckan/ckan-instances/>

Value

(character) A character vector of CKAN server URLs.

Examples

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

# what version is each CKAN server running
out <- lapply(servers()[1:6], function(w) {
  ckan_version(w)$version_num
})

## End(Not run)
```

src_ckan

*Connect to CKAN with dplyr***Description**

Use `src_ckan` to connect to an existing CKAN instance. Use `tbl` to connect to tables in that CKAN through the DataStore Data API.

Usage

```
src_ckan(url, key = get_default_key())
```

Arguments

<code>url</code>	The url of the CKAN instance
<code>key</code>	An optional CKAN API key

Value

A `src_ckan` object wrapping a `CKANConnection`; use it with `dplyr::tbl()` to query DataStore tables.

Examples

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

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

# The primary dbplyr interface uses the DBI connection directly
con <- my_ckan$con
dplyr::tbl(con, "resource-id") |>
  dplyr::filter(status == "active") |>
  dplyr::collect()

# List all tables in the CKAN instance
DBI::dbListTables(con)

# `src_ckan()` remains available for existing code
my_tbl <- dplyr::tbl(
  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")

# The DataStore interface is read-only. `collect()` retrieves the result
```

```
# from CKAN. Filter or limit large tables before collecting them.

## End(Not run)
```

tag_autocomplete	<i>Autocomplete tag names.</i>
------------------	--------------------------------

Description

Return a list of tag names that contain a query string. See <https://docs.ckan.org/en/2.12/api/> for the official API contract.

Usage

```
tag_autocomplete(
  q,
  vocabulary_id = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)
```

Arguments

<code>q</code>	(character) Partial string to search for. The function returns entries whose name or title contains this string. (required)
<code>vocabulary_id</code>	(character) The function restricts matches to this vocabulary.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
tag_autocomplete(q = "ckan")

## End(Not run)
```

tag_create

*Create a tag***Description**

You must be a sysadmin to create vocabulary tags.

Usage

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

Arguments

name	(character) Name for the new tag. The name is a string between 2 and 100 characters long. The name contains only alphanumeric characters and -, _ and .. For example, 'Jazz'.
vocabulary_id	(character) ID of the vocabulary. You add the new tag to this vocabulary. For example, the ID of vocabulary 'Genre'.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the as parameter.

Examples

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

## End(Not run)
```

tag_delete	Delete a tag.
------------	---------------

Description

Delete a tag. You must be a sysadmin to delete tags. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.delete.tag_delete for the official API contract.

Usage

```
tag_delete(  
  id,  
  vocabulary_id = NULL,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)
```

Arguments

id	(character or ckan_tag) The id or name of the tag to delete, or a ckan_tag object.
vocabulary_id	(character) The id or name of the vocabulary that the tag belongs to. Omit it (default NULL) for free tags (optional).
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.delete.tag_delete

Examples

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

# create a vocabulary and a tag in it
vocab <- vocabulary_create(name = "delete-me-vocab")
tag <- tag_create(name = "delete-me-tag", vocabulary_id = vocab$id)

# delete the tag, then clean up the vocabulary
tag_delete(tag$id, vocabulary_id = vocab$id)
vocabulary_delete(vocab$id)

## 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) Tag name query to search for. If you give a query, the function returns only tags whose names contain this string.
vocabulary_id	(character) ID or name of a vocabulary. If you give a vocabulary, the function returns only tags that belong to this vocabulary.
all_fields	(logical) The function returns full tag dictionaries instead of names. Default is FALSE.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses `jsonlite::fromJSON(..., simplifyDataFrame = TRUE)`, which attempts to parse data to `data.frame`'s when possible. The result can vary from a vector, list or `data.frame`. (required)

... Extra curl arguments. The function passes them to [verb-POST](#) (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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	<i>Search tags.</i>
------------	---------------------

Description

Search tags.

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) Tag name query to search for. If you give a query, the function returns only tags whose names contain this string. The value is one or more search strings.
vocabulary_id	(character) ID or name of a vocabulary. If you give a vocabulary, the function returns only tags that belong to this vocabulary.
offset	(numeric) Where to start getting activity items from (optional, default: 0)
limit	(numeric) The maximum number of activities to return (optional, default: 31)
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
tag_search(query = "ta")
tag_search(query = c("ta", "al"))

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

## End(Not run)
```

tag_show	<i>Show a tag.</i>
----------	--------------------

Description

Show a tag.

Usage

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

Arguments

<code>id</code>	(character) Name or ID of the tag.
<code>include_datasets</code>	Include a list of up to 1000 datasets of the tag. The limit is 1000 datasets. Use package_search() for more. Optional. Default is FALSE.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

# show a tag
(x <- tag_show(tags[[30]]$id))

# give back different data formats
tag_show(tags[[30]]$id, as = "json")
tag_show(tags[[30]]$id, as = "table")

## End(Not run)
```

task_status	<i>CKAN task status maintenance</i>
-------------	-------------------------------------

Description

These helpers wrap the sysadmin-only task status endpoints that CKAN uses to track background operations (for example DataPusher runs).

Usage

```
task_status_show(  
    id = NULL,  
    entity_id = NULL,  
    entity_type = NULL,  
    task_type = NULL,  
    task_key = NULL,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)
```

```
task_status_update(  
    id = NULL,  
    entity_id = NULL,  
    entity_type = NULL,  
    task_type = NULL,  
    task_key = NULL,  
    value = NULL,  
    state = NULL,  
    last_updated = NULL,  
    error = NULL,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)
```

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

```
task_status_delete(  
    ...  
)
```

```

    id = NULL,
    entity_id = NULL,
    entity_type = NULL,
    task_type = NULL,
    task_key = NULL,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)

```

Arguments

id	(character) Task status identifier to inspect or mutate.
entity_id	(character) ID of the dataset/resource related to the task.
entity_type	(character) CKAN domain object type (for example "dataset").
task_type	(character) Task namespace such as "datapusher".
task_key	(character) Additional task key (for example queue name).
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)
value	(character) Free-form value, often notes or payload references.
state	(character) Task state such as "queued", "running", or "finished".
last_updated	(character) Timestamp string for the last state change.
error	(character) Optional serialized error blob attached to the task.
data	(list) For <code>task_status_update_many()</code> , a list of task status dictionaries to update.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-ckan-key")
task_status_show(entity_type = "dataset", entity_id = "my-dataset")
task_status_update(
  entity_id = "my-dataset",
  entity_type = "dataset",
  task_type = "datapusher",

```

```

    task_key = "default",
    value = "queued",
    state = "running"
)

## End(Not run)

```

term_translation	<i>Term translation helpers</i>
------------------	---------------------------------

Description

Manage localized UI strings stored in CKAN's term translation table.

Usage

```

term_translation_show(
  terms,
  lang_codes = NULL,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

term_translation_update(
  term,
  term_translation,
  lang_code,
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

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

```

Arguments

terms	(character) Vector of source terms.
lang_codes	(character) Language codes to include.

url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)
term	(character) Source term.
term_translation	(character) Translation value.
lang_code	(character) Target language code.
data	(list) List of translation dictionaries with term, term_translation, and lang_code entries.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```
## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = "my-ckan-key")
term_translation_show(c("License", "Dataset"), lang_codes = "fr")
term_translation_update(term = "License", term_translation = "Licence", lang_code = "fr")

## End(Not run)
```

user_activity_list	<i>Return a list of a user's activities</i>
--------------------	---

Description

Return a list of a user's activities

Usage

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

Arguments

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

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

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

# list package activity
user_activity_list("sckotttie")

# input a ckan_user object
(x <- user_show("sckotttie"))
user_activity_list(x)

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

## End(Not run)
```

<code>user_autocomplete</code>	<i>Autocomplete user names.</i>
--------------------------------	---------------------------------

Description

Return a list of user names that contain a query string. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.user_autocomplete for the official API contract.

Usage

```
user_autocomplete(  
  q,  
  limit = 20,  
  url = get_default_url(),  
  key = get_default_key(),  
  as = "list",  
  ...  
)
```

Arguments

q	(character) Partial string to search for. The function returns entries whose name or title contains this string. (required)
limit	(numeric) The maximum number of user names to return. Default: 20 (optional)
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list of user dictionaries, each with the keys name, fullname, and id. With `as = "table"`, a `data.frame`; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.get.user_autocomplete

Examples

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

user_create

Create a user.

Description

Create a user.

Usage

```
user_create(
    name,
    email,
    password,
    id = NULL,
    fullname = NULL,
    about = NULL,
    openid = NULL,
    image_url = NULL,
    plugin_extras = NULL,
    with_apitoken = NULL,
    url = get_default_url(),
    key = get_default_key(),
    as = "list",
    ...
)
```

Arguments

name	(character) Name of the new user. The name is a string between 2 and 100 characters in length. The name contains only lowercase alphanumeric characters, - and _. Required.
email	(character) Email address for the new user. Required.
password	(character) Password of the new user. The password is a string of at least 4 characters. Required.
id	(character) ID of the new user. Optional. Plugins may set user IDs at creation (CKAN 2.12+, https://github.com/ckan/ckan/pull/8609).
fullname	(character) Full name of the user. Optional.
about	(character) Description of the new user. Optional.
openid	(character) OpenID of the new user. Optional.
image_url	(character) URL to an image displayed on the user's page. Optional.
plugin_extras	(list) Private extra user data for plugins. Only sysadmins may set this. Plugins should namespace extras with the plugin name. Optional.
with_apitoken	(logical) Create an API token for the user. Optional.

url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

References

https://docs.ckan.org/en/2.12/api/index.html#ckan.logic.action.create.user_create

Examples

```
## Not run:
# Setup
ckanr_setup(
  url = "https://data-demo.dpaw.wa.gov.au",
  key = "824e7c50-9577-4bfa-bf32-246ebed1a8a2"
)

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

## End(Not run)
```

user_delete

Delete a user.

Description

Delete a user.

Usage

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

Arguments

id	(character) ID of the user. Required.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

(logical) TRUE when the function deletes the user successfully.

References

https://docs.ckan.org/en/2.12/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-bf32-246ebed1a8a2"
)

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

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

## End(Not run)
```

user_invite

Invite a user to a group

Description

Invite a user to a group

Usage

```

user_invite(
  email,
  group_id,
  role = "member",
  url = get_default_url(),
  key = get_default_key(),
  as = "list",
  ...
)

```

Arguments

email	(character) Email address for the invitee.
group_id	(character or ckan_group) Group identifier.
role	(character) Desired role ("member", "editor", "admin").
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

## Not run:
ckanr_setup(url = "https://demo.ckan.org/", key = getOption("ckan_demo_key"))
user_invite("user@example.com", group_id = "my-group", role = "member")

## End(Not run)

```

user_list

Return a list of the site's user accounts.

Description

Return a list of the site's user accounts.

Usage

```

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

```

Arguments

<code>q</code>	(character) String to filter user names. If you give a string, the function returns only users whose names contain the string.
<code>order_by</code>	(character) Field to sort the list. Optional. Default is 'name'.
<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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_patch	<i>Patch a user account.</i>
------------	------------------------------

Description

This function partially updates a user account: it updates only the provided parameters and leaves all other parameters unchanged (unlike `user_update()`, which may delete parameters not explicitly provided). Normal users can only patch their own user accounts; sysadmins can patch any user account. See https://docs.ckan.org/en/2.11/api/#ckan.logic.action.patch.user_patch for the official API contract.

Usage

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

Arguments

x	(list) A list with key-value pairs
id	(character or ckan_user) The id or name of the user to patch, or a ckan_user object.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

The patched user account as a ckan_user object. With `as = "table"`, a data.frame; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.patch.user_patch

Examples

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

# Create a user, then patch it
usr <- user_create(
  name = "stacy-patch", email = "stacy-patch@example.com",
  password = "helloworld"
)
user_patch(list(about = "patched via ckanr"), id = usr)

# Clean up
user_delete(usr$id)

## 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) User identifier.
user_obj	(user dictionary) User dictionary of the user. Optional.
include_datasets	(logical) Include a list of datasets that the user creates. If the same user or a sysadmin requests the data, the function includes datasets that are draft or private. Optional. Default is False. The limit is 50.
include_num_followers	(logical) Include the number of followers of the user. Optional. Default is False.

url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

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)
```

user_update	<i>Update a user account.</i>
-------------	-------------------------------

Description

Normal users can only update their own user accounts. Sysadmins can update any user account and modify existing usernames. Update methods may delete parameters not explicitly provided: if you want to edit only specific attributes, use [user_patch\(\)](#) instead. For the full list of accepted fields, see [user_create\(\)](#) and https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.user_update.

Usage

```
user_update(
  x,
  id,
  url = get_default_url(),
  key = get_default_key(),
```

```

    as = "list",
    ...
  )

```

Arguments

x	(list) A list with key-value pairs
id	(character or ckan_user) The name or id of the user to update, or a ckan_user object.
url	Base URL to use. Default: https://demo.ckan.org/ . 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 the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to data.frame's when possible. The result can vary from a vector, list or data.frame. (required)
...	Extra curl arguments. The function passes them to verb-POST (optional)

Value

The updated user account as a ckan_user object. With `as = "table"`, a data.frame; with `as = "json"`, the raw JSON response.

References

https://docs.ckan.org/en/2.11/api/#ckan.logic.action.update.user_update

Examples

```

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

# Create a user, then update it (user_update replaces the whole account,
# so fetch the full object first and modify it)
usr <- user_create(
  name = "stacy-update", email = "stacy-update@example.com",
  password = "helloworld"
)
full <- unclass(user_show(usr$id))
full$fullname <- "Stacy Updated"
user_update(full, id = usr)

# Clean up
user_delete(usr$id)

## End(Not run)

```

vocabulary*Manage CKAN vocabularies.*

Description

CKAN allows only sysadmin users to create, update, and delete vocabularies. Each helper wraps the matching `/api/3/action/vocabulary_*` endpoint.

Usage

```
vocabulary_list(  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
vocabulary_show(  
    id,  
    include_datasets = FALSE,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
vocabulary_create(  
    name,  
    tags = NULL,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
vocabulary_update(  
    id,  
    name = NULL,  
    tags = NULL,  
    url = get_default_url(),  
    key = get_default_key(),  
    as = "list",  
    ...  
)  
  
vocabulary_delete(  
    id,
```

```

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

```

Arguments

<code>url</code>	Base URL to use. Default: https://demo.ckan.org/ . See also ckanr_setup and get_default_url .
<code>key</code>	A privileged CKAN API key. Default: your key set with ckanr_setup
<code>as</code>	(character) One of list (default), table, or json. Parsing with the table option uses <code>jsonlite::fromJSON(..., simplifyDataFrame = TRUE)</code> , which attempts to parse data to <code>data.frame</code> 's when possible. The result can vary from a vector, list or <code>data.frame</code> . (required)
<code>...</code>	Extra curl arguments. The function passes them to verb-POST (optional)
<code>id</code>	(character or <code>ckan_vocabulary</code>) Vocabulary ID or name. You can also pass an existing <code>ckan_vocabulary</code> object.
<code>include_datasets</code>	(logical) The function returns datasets owned by the vocabulary. Only applies to <code>vocabulary_show()</code> .
<code>name</code>	(character) Unique name of the vocabulary.
<code>tags</code>	(list) Optional list of tag objects. Each object contains at least a name field. See https://docs.ckan.org/en/2.12/api/ for the full structure.

Value

A list (default), table, or JSON string with the CKAN response, controlled by the `as` parameter.

Examples

```

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

vocab_name <- sprintf("demo_vocab_%s", sample.int(1e6, 1))
vocab <- vocabulary_create(name = vocab_name)

vocabulary_show(vocab$id)
vocabulary_update(vocab, name = paste0(vocab_name, "_updated"))
vocabulary_list()
vocabulary_delete(vocab)

## End(Not run)

```

Index

- * **ckanr settings**
 - ckanr_settings, 30
- * **package**
 - ckanr-package, 5
- activity_create (activity_helpers), 6
- activity_data_show (activity_helpers), 6
- activity_delete (activity_purge), 9
- activity_delete_all (activity_purge), 9
- activity_delete_counts
 - (activity_purge), 9
- activity_diff (activity_helpers), 6
- activity_helpers, 6
- activity_purge, 9
- activity_show (activity_helpers), 6
- am_following_user (follow_users), 52
- api_token_create (api_tokens), 11
- api_token_list (api_tokens), 11
- api_token_revoke (api_tokens), 11
- api_tokens, 11
- as.ckan_file, 12
- as.ckan_file(), 26
- as.ckan_group, 13
- as.ckan_organization, 14
- as.ckan_package, 15
- as.ckan_package(), 26
- as.ckan_related, 16
- as.ckan_related(), 26
- as.ckan_resource, 17
- as.ckan_resource(), 26
- as.ckan_resource_view, 18
- as.ckan_resource_view(), 26
- as.ckan_tag, 19
- as.ckan_user, 20
- as.ckan_vocabulary, 21
- bulk_update, 21
- bulk_update_delete (bulk_update), 21
- bulk_update_private (bulk_update), 21
- bulk_update_public (bulk_update), 21
- changes, 23
- ckan_action, 24
- ckan_classes, 25
- ckan_fetch, 26
- ckan_info, 29
- ckan_version (ckan_info), 29
- ckan_version(), 121
- ckanr (ckanr-package), 5
- ckanr-deprecated, 30
- ckanr-package, 5
- ckanr_settings, 30
- ckanr_setup, 8, 10, 11, 22–24, 31, 34–37, 39, 41, 42, 44, 46, 47, 50, 52, 53, 55–58, 60–63, 65–67, 69–76, 78–84, 86–92, 94–97, 99–103, 105–111, 113–119, 121, 123–125, 127–130, 132, 134, 136–138, 140, 143–150, 153–156, 158, 159, 161, 163–165, 167–171, 173, 174, 176
- ckanr_setup(), 5, 24, 30–32, 66, 107, 128, 134
- config_option_list (config_options), 33
- config_option_show (config_options), 33
- config_option_update (config_options), 33
- config_options, 33
- crul::HttpClient, 24
- crul::proxy(), 32
- crul::verb-GET, 30
- crul::verb-POST, 66, 107, 128, 134
- dashboard_activity_list, 35
- dashboard_count, 36
- dashboard_mark_activities_old
 - (activity_helpers), 6
- dashboard_new_activities_count
 - (activity_helpers), 6
- dataset_am_following (follow_objects), 51
- dataset_follow (follow_objects), 51

- dataset_follower_count
 - (follower_counts), 54
- dataset_follower_list (follower_lists), 55
- dataset_follower_count
 - (follower_counts), 57
- dataset_follower_list (follower_lists), 59
- dataset_purge, 37
- dataset_unfollow (follow_objects), 51
- datastore_extra, 37
- diagnostics, 40
- dplyr-interface (src_ckan), 152
- ds_create, 41
- ds_create(), 5
- ds_create_dataset, 43
- ds_create_dataset(), 5, 30
- ds_delete (datastore_extra), 37
- ds_function_create (datastore_extra), 37
- ds_function_delete (datastore_extra), 37
- ds_info (datastore_extra), 37
- ds_records_delete (datastore_extra), 37
- ds_run_triggers (datastore_extra), 37
- ds_search, 45
- ds_search(), 5
- ds_search_sql, 47
- ds_search_sql(), 5
- ds_upsert (datastore_extra), 37
- ds_upsert(), 42

- file_create (files), 48
- file_delete (files), 48
- file_owner_scan (files), 48
- file_ownership_transfer (files), 48
- file_pin (files), 48
- file_register (files), 48
- file_rename (files), 48
- file_show (files), 48
- file_show(), 12
- file_unpin (files), 48
- files, 48
- follow_objects, 51
- follow_user (follow_users), 52
- follow_users, 52
- follower_count (follower_counts), 54
- follower_counts, 54
- follower_list (follower_lists), 55
- follower_lists, 55
- follower_counts, 57
- follower_lists, 59
- format_autocomplete, 60
- fromJSON, 27

- get_default_key (ckanr_settings), 30
- get_default_key(), 5, 31
- get_default_url, 8, 10, 11, 22, 23, 34–37, 39, 41, 42, 44, 46, 47, 50, 52, 53, 55, 56, 58, 60–63, 65, 67, 69–71, 73–76, 78–84, 86–92, 94–97, 99–103, 105, 106, 108–111, 113–119, 121, 123–125, 127, 129, 130, 132, 134, 136–138, 140, 143–150, 153–156, 158, 159, 161, 163–165, 167–171, 173, 174, 176
- get_default_url (ckanr_settings), 30
- get_default_url(), 24, 30, 31, 66, 107, 128, 134
- get_site_user, 61
- get_test_behaviour (ckanr_settings), 30
- get_test_did (ckanr_settings), 30
- get_test_did(), 31
- get_test_gid (ckanr_settings), 30
- get_test_gid(), 31
- get_test_key (ckanr_settings), 30
- get_test_key(), 31
- get_test_oid (ckanr_settings), 30
- get_test_rid (ckanr_settings), 30
- get_test_rid(), 31
- get_test_url (ckanr_settings), 30
- get_test_url(), 31
- group_activity_list (activity_helpers), 6
- group_am_following (follow_objects), 51
- group_autocomplete, 63
- group_create, 64
- group_delete, 66
- group_follow (follow_objects), 51
- group_follower_count (follower_counts), 54
- group_follower_list (follower_lists), 55
- group_follower_count (follower_counts), 57
- group_follower_list (follower_lists), 59
- group_list, 67
- group_list(), 5, 65, 105
- group_list_authz, 68
- group_member_create, 69
- group_member_delete, 70

- group_package_show, 71
- group_patch, 72
- group_purge, 73
- group_show, 75
- group_show(), 13
- group_unfollow (follow_objects), 51
- group_update, 76
- help_show (diagnostics), 40
- is.ckan_file (as.ckan_file), 12
- is.ckan_group (as.ckan_group), 13
- is.ckan_organization
 - (as.ckan_organization), 14
- is.ckan_package (as.ckan_package), 15
- is.ckan_related (as.ckan_related), 16
- is.ckan_resource (as.ckan_resource), 17
- is.ckan_resource_view
 - (as.ckan_resource_view), 18
- is.ckan_tag (as.ckan_tag), 19
- is.ckan_user (as.ckan_user), 20
- is.ckan_vocabulary
 - (as.ckan_vocabulary), 21
- job_cancel (job_queue), 77
- job_clear (job_queue), 77
- job_list (job_queue), 77
- job_queue, 77
- job_show (job_queue), 77
- license_list, 79
- member_create, 80
- member_delete, 81
- member_list, 82
- member_roles_list, 83
- membership, 83
- organization_activity_list
 - (activity_helpers), 6
- organization_autocomplete, 84
- organization_create, 85
- organization_create(), 5, 96
- organization_delete, 87
- organization_followee_count
 - (followee_counts), 54
- organization_followee_list
 - (followee_lists), 55
- organization_follower_count
 - (follower_counts), 57
- organization_follower_list
 - (follower_lists), 59
- organization_list, 88
- organization_list(), 5, 105
- organization_list_for_user, 89
- organization_member_create, 90
- organization_member_delete, 91
- organization_patch, 92
- organization_patch(), 96
- organization_purge, 93
- organization_show, 94
- organization_show(), 5, 14
- organization_update, 96
- organization_update(), 92
- package_activity_list, 97
- package_autocomplete, 98
- package_collaborator_create, 99
- package_collaborator_delete, 100
- package_collaborator_list, 101
- package_collaborator_list_for_user,
 - 102
- package_collaborators, 103
- package_create, 103
- package_create(), 5
- package_create_default_resource_views,
 - 106
- package_dataset_extras, 107
- package_delete, 107
- package_list, 108
- package_list_current, 109
- package_owner_org_update, 110
- package_patch, 111
- package_patch(), 110, 124
- package_relationship_create, 112
- package_relationship_create(), 116
- package_relationship_delete, 113
- package_relationship_update, 114
- package_relationships, 115
- package_relationships_list, 116
- package_resource_reorder, 117
- package_revise, 118
- package_revise(), 124
- package_revision_list, 119
- package_search, 120
- package_search(), 159
- package_show, 122
- package_show(), 15
- package_update, 124

- ping, 125
- read.csv, 27
- read.csv(), 5
- read.table, 27
- read_excel, 27
- read_html, 27
- read_parquet, 27
- read_xml, 27
- recently_changed_packages_activity_list
 (activity_helpers), 6
- related_create, 126
- related_delete, 127
- related_list, 128
- related_show, 129
- related_show(), 16
- resource_create, 131
- resource_create(), 5, 30, 42, 44, 105
- resource_create_default_resource_views,
 133
- resource_delete, 134
- resource_patch, 135
- resource_search, 137
- resource_show, 138
- resource_show(), 17
- resource_update, 139
- resource_view_clear, 143
- resource_view_create, 144
- resource_view_delete, 145
- resource_view_list, 146
- resource_view_reorder, 147
- resource_view_show, 148
- resource_view_show(), 18
- resource_view_update, 149
- resource_views, 150
- revision_list, 150
- send_email_notifications
 (activity_helpers), 6
- servers, 151
- servers(), 6
- src_ckan, 152
- st_read, 27
- status_show (diagnostics), 40
- tag_autocomplete, 153
- tag_create, 154
- tag_create(), 105
- tag_delete, 155
- tag_list, 156
- tag_list(), 5
- tag_search, 157
- tag_show, 158
- tag_show(), 19
- task_status, 160
- task_status_delete (task_status), 160
- task_status_show (task_status), 160
- task_status_update (task_status), 160
- task_status_update_many (task_status),
 160
- term_translation, 162
- term_translation_show
 (term_translation), 162
- term_translation_update
 (term_translation), 162
- term_translation_update_many
 (term_translation), 162
- unfollow_user (follow_users), 52
- user_activity_list, 163
- user_autocomplete, 164
- user_create, 166
- user_create(), 173
- user_delete, 167
- user_follower_count (follower_counts),
 54
- user_follower_list (follower_lists), 55
- user_follower_count (follower_counts),
 57
- user_follower_list (follower_lists), 59
- user_invite, 168
- user_list, 169
- user_list(), 5
- user_patch, 171
- user_patch(), 173
- user_show, 172
- user_show(), 20
- user_update, 173
- user_update(), 171
- vocabulary, 175
- vocabulary_create (vocabulary), 175
- vocabulary_delete (vocabulary), 175
- vocabulary_list (vocabulary), 175
- vocabulary_show (vocabulary), 175
- vocabulary_update (vocabulary), 175