Package: rhdx (via r-universe)

September 25, 2024

Title A Client for the Humanitarian Data Exchange platform API

Description Client for the Humanitarian Data Exchange (HDX) platform API <https://data.humdata.org>. Allows to search and download HDX datasets into R.

Version 0.1.0.9000

License MIT + file LICENSE

LazyData TRUE

URL https://gitlab.com/dickoa/rhdx

BugReports https://gitlab.com/dickoa/rhdx/issues

Depends R (>= 3.5)

Imports crul (>= 1.1), magrittr, jsonlite, yaml, readr, readxl, sf, stars, memoise, hoardr, tibble, base64enc, countrycode

Suggests roxygen2, knitr, rmarkdown, testthat, covr, webmockr, vcr

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Encoding UTF-8

Language en-US

VignetteBuilder knitr

X-schema.org-applicationCategory Data Access

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

Repository https://humanitarian-user-group.r-universe.dev

RemoteUrl https://github.com/dickoa/rhdx

RemoteRef HEAD

RemoteSha c443336b2d9e2d57dc6c0f6a81ab7bc6db3a631c

Contents

.list_datasets *List datasets*

Description

List datasets

Usage

```
.list_datasets(limit = NULL, offset = NULL, configuration = NULL)

list_datasets(limit = NULL, offset = NULL, configuration = NULL)
```

Arguments

`limit`	integer; limit
`offset`	integer; offset
`configuration`	a Configuration object

Value

A vector of datasets names

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 list_datasets(limit = 10L)

## End(Not run)
```

`.list_organizations` *List HDX organization*

Description

List HDX organization

Usage

```
.list_organizations(
  sort = "name asc",
  all_fields = FALSE,
  include_dataset_count = TRUE,
  include_groups = FALSE,
  include_user = FALSE,
  include_tags = FALSE,
  configuration = NULL,
  ...
)

list_organizations(
  sort = "name asc",
  all_fields = FALSE,
  include_dataset_count = TRUE,
  include_groups = FALSE,
  include_user = FALSE,
  include_tags = FALSE,
  configuration = NULL,
  ...
)
```

Arguments

`sort`	Character how to sort the results. Default is "name asc"
`all_fields`	Logical, include all fields
`include_dataset_count`	Logical include count in the result
`include_groups`	Logical, whether or not to include locations
`include_user`	Logical, whether or not to include user
`include_tags`	Logical whether or not to include tags
`configuration`	Configuration
`...`	extra paramaters

Value

A list of organizations on HDX

`authorized_tags` *List all authorized tags in HDX*

Description

List all authorized tags in HDX

Usage

```
authorized_tags(configuration = NULL)
```

Arguments

`configuration`	Configuration

Value

A vector of character, the authorized tags name

Examples

```
## Not run:
autorized_tags()

## End(Not run)
```

`browse.Dataset` *Browse a HDX object*

Description

Browse a HDX object

Usage

```
## S3 method for class 'Dataset'
browse(x, ...)

## S3 method for class 'Organization'
browse(x, ...)

## S3 method for class 'HDXResource'
browse(x, ...)

## S3 method for class 'HDXUser'
browse(x, ...)
browse(x, ...)

## Default S3 method:
browse(x, ...)
```

Arguments

x	an HDX object
...	Extra parameters

Value

Character Tags of the dataset

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- search_dataset(rows = 3L)
 browse(res[[1]])

## End(Not run)
```

---

count_datasets *Count all datasets on HDX*

Description

Count all datasets on HDX

Usage

```
count_datasets(configuration = NULL)
```

Arguments

configuration	an HDX Configuration object

Value

An integer, the number of datasets

create_rhdx_config *Create an HDX configuration object*

Description

Create and HDX configuration object

Usage

```
create_rhdx_config(
  hdx_site = "prod",
  hdx_key = NULL,
  read_only = TRUE,
  hdx_config = NULL,
  hdx_config_file = NULL
)
```

Arguments

`hdx_site` Character to specify which HDX server you want to use. Default to "prod".

`hdx_key` Character for the CKAN API key, it is required to push data into HDX

`read_only` Logical if `FALSE` and hdx_key provided is correct you can push metdata and data to HDX

`hdx_config` List of HDX configuration

`hdx_config_file` Character, path of the HDX config file in JSON and YAML format

Value

An HDX Configuration object

delete_resource *Delete resource from dataset*

Description

Delete resource from dataset

Usage

```
delete_resource(dataset, index)
```

Arguments

`dataset`	Dataset the dataset from which we one to remove the resource
`index`	Integer the index of the resource to be removed

Details

Delete resource from dataset

Value

Dataset the dataset without the resource

`delete_resources` *Delete all resource from dataset*

Description

Delete all resource from dataset

Usage

```
delete_resources(dataset)
```

Arguments

`dataset`	A Dataset, the dataset to remove

Details

Delete all resources from dataset

Value

Dataset without resources

delete_rhdx_config *Delete rhdx config*

Description

Delete the configuration settings for using rhdx.

Usage

```
delete_rhdx_config()
```

Details

Delete HDX config

Value

None

Examples

```
## Not run:
# Setting the config to use HDX default server
set_rhdx_config(hdx_site = "prod")
get_rhdx_config()

delete_rhdx_config()
get_rhdx_config()

## End(Not run)
```

download_resource *Download an HDX resource*

Description

Download an HDX resource into a specific folder

Usage

```
download_resource(
  resource,
  folder = NULL,
  filename = NULL,
  quiet = FALSE,
  force = FALSE,
  ...
)
```

Arguments

resource	Resource, an HDX resource
folder	character, path of the directory where you will store the data
filename	(character), name of the file you will download
quiet	(logical), no progress bar from download (default = FALSE)
force	(logical) force download (default = FALSE)
...	extra paramaters

Value

Resource

Examples

```
## Not run:
#Setting the config to use HDX default server
 res <- read_resource("98aa1742-b5d3-40c3-94c6-01e31ded6e84")
 download_resource(res, folder = "/tmp")

## End(Not run)
```

---

get_dataset_date *Get the Dataset date*

Description

Date of dataset

Usage

```
get_dataset_date(dataset)
```

Arguments

dataset	Dataset

Value

Date, date of the dataset

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- search_dataset(rows = 3L)
 get_dataset_date(res[[1]])

## End(Not run)
```

get_locations_names *Dataset locations name*

Description

Gets locations name from the datasets

Usage

```
get_locations_names(dataset)
```

Arguments

dataset	Dataset

Value

Character locations of the dataset

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- search_dataset(rows = 3L)
 get_location_names(res[[1]])

## End(Not run)
```

get_organization_name *Dataset organization name*

Description

Get the organization sharing the data

Usage

```
get_organization_name(dataset)
```

Arguments

dataset	Dataset

Value

Character The name of the organization sharing the data

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- search_dataset(rows = 3L)
 get_organization_name(res[[1]])

## End(Not run)
```

get_resource *Add resource to dataset*

Description

Add resource to dataset

Usage

```
get_resource(dataset, index)
```

Arguments

`dataset`	Dataset
`index`	integer; resource position in the dataset

Value

Resource

get_resources *Add resource to dataset*

Description

Add resource to dataset

Usage

```
get_resources(dataset, pattern = NULL, format = NULL)
```

Arguments

`dataset`	Dataset
`pattern`	regex pattern in resource name
`format`	format of the resources

Value

resource_list

`get_resources_formats` *Dataset resources format*

Description

Gets format of all resources from the datasets

Usage

```
get_resources_formats(dataset)
```

Arguments

`dataset`	Dataset

Value

Character Format of the resources

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- search_dataset(rows = 3L)
 get_resources_formats(res[[1]])

## End(Not run)
```

`get_resource_dataset` *Get the dataset containing the resource*

Description

Get the dataset containing the resource

Usage

```
get_resource_dataset(resource)
```

Arguments

`resource`	Resource, an HDX resource

Value

a Dataset, the dataset containing the resource

`get_resource_format` *Get the file format of the resource*

Description

Get the file format of the resource

Usage

```
get_resource_format(resource)
```

Arguments

`resource`	Resource, an HDX resource

Value

A character, the format of the resource

`get_resource_layers` *List layers available in spatial resources on HDX*

Description

List layers available in spatial resources on HDX

Usage

```
get_resource_layers(
  resource,
  format = NULL,
  download_folder = NULL,
  quiet = TRUE
)
```

Arguments

`resource`	Resource, an HDX resource
`format`	character; file format
`download_folder`	Character, path of the directory where you will store the data
`quiet`	Logical, no progress bar from download (default = FALSE)

Value

the layers name

get_resource_sheets *Get the names of the sheets of XLS(X) resources*

Description

Get the names of the sheets of XLS(X) resources

Usage

```
get_resource_sheets(
  resource,
  format = NULL,
  download_folder = NULL,
  quiet = TRUE
)
```

Arguments

resource	Resource, an HDX resource
format	character; file format
download_folder	character, path of the directory where you will store the data
quiet	logical, no progress bar from download (default = FALSE)

Value

the names of the sheets of XLS(X) resources

get_tags_names *Dataset tags name*

Description

Gets dataset tags name

Usage

```
get_tags_names(dataset)
```

Arguments

dataset	Dataset

Value

Character Tags of the dataset

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- search_dataset(rows = 3L)
 get_tags_names(res[[1]])

## End(Not run)
```

HDXConfig *HDX Configuration*

Description

HDX Configuration

HDX Configuration

Details

HDX Configuration allow to connect to an HDX server and setup project where you can interact with the HDX platform

Public fields

`data` all info in list.

Methods

Public methods:

- `HDXConfig$new()`
- `HDXConfig$get_credentials()`
- `HDXConfig$set_read_only()`
- `HDXConfig$set_hdx_key()`
- `HDXConfig$get_hdx_key()`
- `HDXConfig$set_hdx_site()`
- `HDXConfig$get_hdx_site()`
- `HDXConfig$get_hdx_site_url()`
- `HDXConfig$remoteclient()`
- `HDXConfig$call_action()`
- `HDXConfig$read()`
- `HDXConfig$setup()`

- `HDXConfig$delete()`
- `HDXConfig$get_global_config()`
- `HDXConfig$general_statistics()`
- `HDXConfig$as_list()`
- `HDXConfig$print()`
- `HDXConfig$clone()`

Method `new()`**:** Create a new Configuration object.

Usage:

```
HDXConfig$new(
  hdx_site = "prod",
  hdx_key = NULL,
  hdx_config = NULL,
  hdx_config_file = NULL,
  read_only = TRUE,
  user_agent = NULL
)
```

Arguments:

`hdx_site` character the server instance to use

`hdx_key` character, the HDX API key

`hdx_config` configuration in a list

`hdx_config_file` a character value config file. default is the config supplied in the package

`read_only` a logical value indicating if you want to just read or be also able to write on the HDX server. You will need a API key to write.

`user_agent` a character value, User agent

Returns: A new Configuration object.

Method `get_credentials()`**:** Configuration credentials when using a HDX API key

Usage:

```
HDXConfig$get_credentials()
```

Returns: the username and password associated to the HDX API key

Method `set_read_only()`**:** Create or revoke read only status

Usage:

```
HDXConfig$set_read_only(read_only = TRUE)
```

Arguments:

`read_only` a logical value indicating if you want to just read or be also able to write on the HDX server. You will need a API key to write.

Method `set_hdx_key()`**:** Specify a HDX API key

Usage:

```
HDXConfig$set_hdx_key(hdx_key)
```

Arguments:

`hdx_key` a character with key

Method `get_hdx_key()`**:** Specify a HDX API key

Usage:

```
HDXConfig$get_hdx_key()
```

Returns: a character, the HDX API key

Method `set_hdx_site()`**:** Specify a HDX server to use

Usage:

```
HDXConfig$set_hdx_site(hdx_site = "prod")
```

Arguments:

`hdx_site` a character, the server type to use, `prod`, `test`, `feature` or `demo`

Returns: a character, the HDX API key

Method `get_hdx_site()`**:** Get the HDX server in use

Usage:

```
HDXConfig$get_hdx_site()
```

Returns: the server type

Method `get_hdx_site_url()`**:** Get the HDX server URL in use

Usage:

```
HDXConfig$get_hdx_site_url()
```

Returns: the server URL

Method `remoteclient()`**:** Get the remoteclient currently used

Usage:

```
HDXConfig$remoteclient()
```

Returns: a crul::HttpClient

Method `call_action()`**:** Call the client to the HDX API

Usage:

```
HDXConfig$call_action(action, ..., verb = "get")
```

Arguments:

`action` a character

`...` parameters for each verb used

`verb` a character the verb used, `post`, `get`, `put` or `patch`

Returns: list a with status code and results

Method `read()`**:** read and show Configuration object

Usage:

```
HDXConfig$read()
```

Returns: Configuration object

Method `setup()`**:** Setup Configuration object

Usage:

```
HDXConfig$setup(
  hdx_site = "prod",
  hdx_key = NULL,
  read_only = TRUE,
  hdx_config = NULL,
  configuration = NULL
)
```

Arguments:

`hdx_site` a character value, the server

`hdx_key` a character value, the API key

`read_only` a logical value read only

`hdx_config` a list

`configuration` a character

Method `delete()`**:** Delete a Configuration object Access the global Configuration

Usage:

```
HDXConfig$delete()
```

Method `get_global_config()`**:**

Usage:

```
HDXConfig$get_global_config()
```

Returns: list with HDX configuration information

Method `general_statistics()`**:** Get general statistics about the server

Usage:

```
HDXConfig$general_statistics()
```

Returns: list with statistics about the server

Method `as_list()`**:** Convert configuration to list

Usage:

```
HDXConfig$as_list()
```

Returns: configuration in list format

Method `print()`**:** Print Configuration object

Usage:

```
HDXConfig$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
set_rhdx_config(hdx_site = "prod")
get_rhd_config()

## End(Not run)
```

HDXDataset *HDX Dataset*

Description

HDX Dataset

HDX Dataset

Details

Dataset class containing all logic for accessing, creating, and updating datasets and associated resources.

Super class

`rhdx::HDXObject` -> `HDXDataset`

Public fields

`resources` list of Resource object within the dataset

`data` placeholder for Dataset field element

Methods

Public methods:

- `HDXDataset$new()`
- `HDXDataset$get_resource()`
- `HDXDataset$get_resources()`
- `HDXDataset$number_of_resources()`
- `HDXDataset$delete_resource()`
- `HDXDataset$delete_resources()`
- `HDXDataset$browse()`
- `HDXDataset$get_configuration()`
- `HDXDataset$get_dataset_date()`
- `HDXDataset$get_update_frequency()`
- `HDXDataset$get_tags()`
- `HDXDataset$get_locations()`
- `HDXDataset$get_maintainer()`

- `HDXDataset$get_organization()`
- `HDXDataset$get_showcases()`
- `HDXDataset$is_requestable()`
- `HDXDataset$get_required_fields()`
- `HDXDataset$check_required_fields()`
- `HDXDataset$as_list()`
- `HDXDataset$print()`
- `HDXDataset$clone()`

Method `new()`**:** Create a new Dataset object

Usage:

```
HDXDataset$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list with required field to create a dataset

`configuration` a Configuration object

Returns: A Dataset object

Method `get_resource()`**:** Get a specific resource of the dataset

Usage:

```
HDXDataset$get_resource(index)
```

Arguments:

`index,` the index of the resource to access

Returns: a Resource object, the selected resource

Method `get_resources()`**:** Get all resources of the dataset

Usage:

```
HDXDataset$get_resources(pattern = NULL, format = NULL)
```

Arguments:

`pattern` regex pattern in resource name

`format` format of the resources

Returns: a list of Resource objects, all resources available in the dataset

Method `number_of_resources()`**:** Get number of dataset resources

Usage:

```
HDXDataset$number_of_resources()
```

Returns: The number of Resource objects

Method `delete_resource()`**:** Delete a resource by its index

Usage:

```
HDXDataset$delete_resource(index = 1L)
```

Arguments:

`index,` the index of the resource to delete

Method `delete_resources()`**:** Delete all resources from a dataset

Usage:

```
HDXDataset$delete_resources()
```

Method `browse()`**:** Browse the dataset page on HDX

Usage:

```
HDXDataset$browse()
```

Method `get_configuration()`**:** Get the current configuration in use

Usage:

```
HDXDataset$get_configuration()
```

Returns: A configuration object, the configuration in use

Method `get_dataset_date()`**:** Get the dataset date

Usage:

```
HDXDataset$get_dataset_date()
```

Returns: a date, the dataset date.

Method `get_update_frequency()`**:** Get dataset update frequency

Usage:

```
HDXDataset$get_update_frequency()
```

Returns: a character, the dataset update frequency Get dataset tags

Method `get_tags()`**:**

Usage:

```
HDXDataset$get_tags()
```

Returns: a list of Tag objects, datasets tags

Method `get_locations()`**:** Get the datasets location

Usage:

```
HDXDataset$get_locations()
```

Returns: a list of Location objects, all locations covered by the dataset

Method `get_maintainer()`**:** Get the dataset maintainer

Usage:

```
HDXDataset$get_maintainer()
```

Returns: An User object, the maintainer of the dataset

Method `get_organization()`**:** Get the dataset organization

Usage:

```
HDXDataset$get_organization()
```

Returns: an Organization object, the organization that shared the data

Method `get_showcases()`**:** Get the Showcase associated to the dataset

Usage:

```
HDXDataset$get_showcases()
```

Returns: a Showcase object containing the dataset

Method `is_requestable()`**:** Check if the dataset is requestable

Usage:

```
HDXDataset$is_requestable()
```

Returns: a logical value, `TRUE` if it's a requestable dataset

Method `get_required_fields()`**:** Get dataset required fields

Usage:

```
HDXDataset$get_required_fields()
```

Returns: list of required fields for a dataset

Method `check_required_fields()`**:** Check dataset required field

Usage:

```
HDXDataset$check_required_fields()
```

Returns: a logical value, TRUE if the the dataset is not missing a required field and throws an error otherwise

Method `as_list()`**:** Get dataset field into list

Usage:

```
HDXDataset$as_list()
```

Returns: a list with dataset field

Method `print()`**:** Print a Dataset object

Usage:

```
HDXDataset$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXDataset$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

HDXLocation *HDX Location*

Description

HDX Location

HDX Location

Details

HDX location mostly countries

Super class

`rhdx::HDXObject` -> `HDXLocation`

Public fields

`data` placeholder location

Methods

Public methods:

- `HDXLocation$new()`
- `HDXLocation$get_required_fields()`
- `HDXLocation$check_required_fields()`
- `HDXLocation$browse()`
- `HDXLocation$as_list()`
- `HDXLocation$print()`
- `HDXLocation$clone()`

Method `new()`**:** Create a new Location object

Usage:

```
HDXLocation$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list with required field to create a dataset

`configuration` a Configuration object

Returns: A Location object

Method `get_required_fields()`**:** Get dataset required fields

Usage:

```
HDXLocation$get_required_fields()
```

Returns: list of required fields for a dataset

Method `check_required_fields()`**:** Check dataset required field

Usage:

```
HDXLocation$check_required_fields()
```

Returns: a logical value, TRUE if the the dataset is not missing a required field and throws an error otherwise

Method `browse()`**:** Browser the Location page on HDX

Usage:

```
HDXLocation$browse()
```

Method `as_list()`**:** Get location field into list

Usage:

```
HDXLocation$as_list()
```

Returns: a list with dataset field

Method `print()`**:** Print a Dataset object

Usage:

```
HDXLocation$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXLocation$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`HDXObject` *HDXObject abstract class*

Description

HDXObject abstract class

HDXObject abstract class

Details

HDXObject class containing all logic for accessing, creating, and updating HDX objects.

Public fields

`data` placeholder for HDXObject field element

Methods

Public methods:

- `HDXObject$new()`
- `HDXObject$update_from_yaml()`
- `HDXObject$update_from_json()`
- `HDXObject$get_required_fields()`
- `HDXObject$check_required_field()`
- `HDXObject$as_list()`
- `HDXObject$browse()`
- `HDXObject$get_configuration()`
- `HDXObject$print()`
- `HDXObject$clone()`

Method `new()`**:** Create a new HDXObject object

Usage:

```
HDXObject$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list with required field to create a HDXObject

`configuration` a Configuration object

Returns: A HDXObject object

Method `update_from_yaml()`**:** Update metadata from yaml file

Usage:

```
HDXObject$update_from_yaml(path)
```

Arguments:

`path` (character) Path to YAML metadata

Method `update_from_json()`**:** Update metadata from json file

Usage:

```
HDXObject$update_from_json(path)
```

Arguments:

`path` (character) Path to JSON metadata

Method `get_required_fields()`**:** Get HDXObject required fields

Usage:

```
HDXObject$get_required_fields()
```

Returns: list of required fields for a resource

Method `check_required_field()`**:** Check HDXobject required field

Usage:

```
HDXObject$check_required_field()
```

Returns: a logical value, TRUE if the the resource is not missing a required field and throws an error otherwise

Method `as_list()`**:** Get HDXObject field into list

Usage:

```
HDXObject$as_list()
```

Returns: a list with HDXObject field

Method `browse()`**:** Browse HDX

Usage:

```
HDXObject$browse()
```

Method `get_configuration()`**:** Get the current configuration in use

Usage:

```
HDXObject$get_configuration()
```

Returns: A configuration object, the configuration in use

Method `print()`**:** Print a Dataset object

Usage:

```
HDXObject$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXObject$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`HDXOrganization` *HDX Organization*

Description

HDX Organization

HDX Organization

Details

HDX Organization

Super class

`rhdx::HDXObject` -> `HDXOrganization`

Public fields

`data` placeholder for the Organization fields element

Methods

Public methods:

- `HDXOrganization$new()`
- `HDXOrganization$get_datasets()`
- `HDXOrganization$browse()`
- `HDXOrganization$as_list()`
- `HDXOrganization$print()`
- `HDXOrganization$clone()`

Method `new()`**:** Create a Organization object

Usage:

```
HDXOrganization$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list with required field to create a dataset

`configuration` a Configuration object

Returns: A Organization object

Method `get_datasets()`**:** Get the list of datasets within the organization

Usage:

```
HDXOrganization$get_datasets()
```

Returns: list of Dataset objects

Method `browse()`**:** Browse the Organization page on HDX

Usage:

```
HDXOrganization$browse()
```

Method `as_list()`**:** Get dataset field into list

Usage:

```
HDXOrganization$as_list()
```

Returns: a list with organization field element

Method `print()`**:** Print a Dataset object

Usage:

```
HDXOrganization$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXOrganization$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`HDXResource` *HDX Resource*

Description

HDX Resource

HDX Resource

Details

HDX Resource, it contains all the logic for creating, checking, and updating resources

Super class

`rhdx::HDXObject` -> `HDXResource`

Public fields

`data` placeholder for Resource field element

Methods

Public methods:

- `HDXResource$new()`
- `HDXResource$download()`
- `HDXResource$download_folder()`
- `HDXResource$read_resource()`
- `HDXResource$get_layers()`
- `HDXResource$get_sheets()`
- `HDXResource$get_dataset()`
- `HDXResource$get_required_fields()`
- `HDXResource$check_required_field()`
- `HDXResource$get_format()`
- `HDXResource$as_list()`
- `HDXResource$browse()`
- `HDXResource$print()`
- `HDXResource$clone()`

Method `new()`**:** Create a new Resource object

Usage:

```
HDXResource$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list with required field to create a resource

`configuration` a Configuration object

Returns: A new Resource object

Method `download()`**:** Download a HDX resource

Usage:

```
HDXResource$download(
  folder = NULL,
  filename = NULL,
  quiet = TRUE,
  force = FALSE,
  ...
)
```

Arguments:

`folder` a character, folder to save the dataset

`filename` a character, filename of the dataset

`quiet` a logical value, silent download if TRUE

`force` a logical value, force download

`...` other `download.file` parameters

Returns: a character, the file path

Method `download_folder()`**:** Get the download folder for the latest downloaded resource

Usage:

```
HDXResource$download_folder()
```

Returns: a character, folder with the latest downloaded resource

Method `read_resource()`**:** Read a Resource object directly into memory

Usage:

```
HDXResource$read_resource(
  sheet = NULL,
  layer = NULL,
  format = NULL,
  download_folder = NULL,
  simplify_json = TRUE,
  force_download = FALSE,
  quiet_download = TRUE,
  ...
)
```

Arguments:

`sheet` a character value, only for resource in Excel format

`layer` a character value, only for spatial (vector) resource

`format` a character value, file format;

`download_folder` a character value, folder to save the downloaded resource

`simplify_json` a logical value

`force_download` a logical value, if TRUE force download

`quiet_download` a logical value, if TRUE silent download

`...` other parameters

Returns: a `tibble`, a `sf`, a `stars` or a `list` depending on the type of resource read

Method `get_layers()`**:** Get spatial (vector) resource list of layers

Usage:

```
HDXResource$get_layers(
  format = NULL,
  download_folder = NULL,
  quiet_download = TRUE,
  force_download = FALSE
)
```

Arguments:

`format` character; file format

`download_folder` a character value, folder to save the downloaded resource

`quiet_download` a logical value, if TRUE silent download

`force_download` a logical value, if TRUE force download

Returns: a the list of layers available in the resource

Method `get_sheets()`**:** Get the list of sheets name of resource

Usage:

```
HDXResource$get_sheets(
  format = NULL,
  download_folder = NULL,
  quiet_download = TRUE,
  force_download = FALSE
)
```

Arguments:

`format` character; file format

`download_folder` a character value, folder to save the downloaded resource

`quiet_download` a logical value, if TRUE silent download

`force_download` a logical value, if TRUE force download

Returns: a the list of layers available in the resource

Method `get_dataset()`**:** Get the resource dataset.

Usage:

```
HDXResource$get_dataset()
```

Returns: a Dataset, the dataset containing the resource

Method `get_required_fields()`**:** Get dataset required fields

Usage:

```
HDXResource$get_required_fields()
```

Returns: list of required fields for a resource

Method `check_required_field()`**:** Check dataset required field

Usage:

```
HDXResource$check_required_field(check_dataset_id = FALSE)
```

Arguments:

`check_dataset_id` logical whether to check or not dataset id

Returns: a logical value, TRUE if the the resource is not missing a required field and throws an error otherwise

Method `get_format()`**:** Get the file format

Usage:

```
HDXResource$get_format()
```

Returns: a character, the file format of the resource

Method `as_list()`**:** Get resource field into list

Usage:

```
HDXResource$as_list()
```

Returns: a list with resource field

Method `browse()`**:** Browse the resource page on HDX

Usage:

```
HDXResource$browse()
```

Method `print()`**:** Print a Resource object

Usage:

```
HDXResource$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXResource$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`HDXShowcase` *HDX Showcase*

Description

HDX Showcase

HDX Showcase

Details

HDX Showcase

Super class

`rhdx::HDXObject` -> `HDXShowcase`

Public fields

`datasets` list of datasets using this showcase

`data` the field info into list

Methods

Public methods:

- `HDXShowcase$new()`
- `HDXShowcase$get_datasets()`
- `HDXShowcase$browse()`
- `HDXShowcase$as_list()`
- `HDXShowcase$print()`
- `HDXShowcase$clone()`

Method `new()`**:** Create a new Showcase object

Usage:

```
HDXShowcase$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list, data with required field to create Showcase

`configuration` Configuration, configuration to use

Returns: a new Showcase object

Method `get_datasets()`**:** List datasets using the Showcase

Usage:

```
HDXShowcase$get_datasets()
```

Returns: a list of dataset

Method `browse()`**:** Browse the Showcase page on HDX

Usage:

```
HDXShowcase$browse()
```

Method `as_list()`**:** Get dataset field into list

Usage:

```
HDXShowcase$as_list()
```

Returns: a list with showcase field info

Method `print()`**:** Print a Showcase object

Usage:

```
HDXShowcase$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXShowcase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`HDXUser` *HDX User*

Description

HDX User

HDX User

Details

HDX user

Super class

`rhdx::HDXObject` -> `HDXUser`

Public fields

`data` placeholder for Dataset field element

Methods

Public methods:

- `HDXUser$new()`
- `HDXUser$as_list()`
- `HDXUser$browse()`
- `HDXUser$print()`
- `HDXUser$clone()`

Method `new()`**:** Create a new

Usage:

```
HDXUser$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list of field required to create a dataset

`configuration` Configuration configuration to use

Returns: a new User object

Method `as_list()`**:** Get dataset field into list

Usage:

```
HDXUser$as_list()
```

Returns: a list with dataset field

Method `browse()`**:** Browse the user page on HDX

Usage:

```
HDXUser$browse()
```

Method `print()`**:** Print a User object

Usage:

```
HDXUser$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HDXUser$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`hdx_general_statistics` *Get general stats about HDX*

Description

Get some stats about HDX

Usage

```
hdx_general_statistics()
```

Value

A list

list_locations *List locations*

Description

List locations

Usage

```
list_locations(
  sort = "name asc",
  all_fields = FALSE,
  configuration = NULL,
  ...
)
```

Arguments

sort	Character sorting of the search results. Default: “name asc”, the allowed fields are ‘name’, ‘package_count’ and ‘title’
all_fields	Logical if TRUE returns list instead of just names
configuration	a Configuration
...	Extra parameters to group_list https://docs.ckan.org/en/ckan-2.8.2/api/index.html#ckan.logic.action.get.g

Value

A vector of locations names

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 list_locations(limit = 10L)

## End(Not run)
```

list_tags *List all tags*

Description

List all available tags

Usage

```
list_tags(
  query = NULL,
  vocabulary_id = NULL,
  all_fields = FALSE,
  configuration = NULL
)
```

Arguments

query	a tag name query to search for, if given only tags whose names contain this string will be returned
vocabulary_id	the id or name of a vocabulary, if give only tags that belong to this vocabulary will be returned
all_fields	logical return full Tag object instead of just names
configuration	Configuration the configuration to use

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 list_tag()

## End(Not run)
```

list_users *List all users*

Description

List all users

Usage

```
list_users(order_by = "number_created_packages", configuration = NULL, ...)
```

Arguments

order_by	Logical user sorted is TRUE
configuration	Configuration the configuration to use
...	Extra parameters

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 list_user()

## End(Not run)
```

list_vocabularies *List available HDX Vocabulary objects*

Description

List available HDX Vocabulary objects

Usage

```
list_vocabularies(identifier = NULL, configuration = NULL)
```

Arguments

identifier	character identifier
configuration	Configuration

Value

A list of Vocabulary object

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- list_vocabularies()
 res

## End(Not run)
```

pull_dataset *Pull HDX dataset into R*

Description

Read an HDX dataset from its name or id

Usage

```
pull_dataset(identifier, configuration = NULL)
```

Arguments

identifier	Character dataset keyword
configuration	a Configuration object

Value

Dataset the dataset

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- pull_dataset("mali-3wop")
 res

## End(Not run)
```

pull_location *Read an HDX location*

Description

Read an HDX location

Usage

```
pull_location(
  identifier = NULL,
  include_datasets = FALSE,
  configuration = NULL,
  ...
)
```

Arguments

`identifier`	Character location uuid
`include_datasets`	Logical whether to include or not dataset
`configuration`	Configuration a configuration object
`...`	Extra parameters

Value

Location

Examples

```
## Not run:
#Setting the config to use HDX default server
 set_rhdx_config()
 res <- pull_location("mli")
 res

## End(Not run)
```

pull_organization *Read an HDX organization*

Description

Read an HDX organization

Usage

```
pull_organization(
  identifier = NULL,
  include_datasets = FALSE,
  configuration = NULL,
  ...
)
```

Arguments

`identifier`	character resource uuid
`include_datasets`	Logical, include datasets if TRUE
`configuration`	an HDX configuration object
`...`	Extra parameters

Value

HDX organization

`pull_resource` *Read an HDX resource*

Description

Read an HDX resource

Usage

```
pull_resource(identifier, configuration = NULL)
```

Arguments

`identifier`	Character resource uuid
`configuration`	a Configuration object

Value

Resource

Examples

```
## Not run:
#Setting the config to use HDX default server
 set_rhdx_config()
 res <- pull_resource("98aa1742-b5d3-40c3-94c6-01e31ded6e84")
 res

## End(Not run)
```

`pull_showcase` *Read Showcase*

Description

Read HDX Showcase

Usage

```
pull_showcase(identifier = NULL, configuration = NULL)
```

Arguments

`identifier`	Character Showcase name or id
`configuration`	Configuration an HDX configuration object

Details

Delete resource from dataset

Value

A showcase

Examples

```
## Not run:
 # Setting the config to use HDX default server
 pull_showcase("fts-requirements-and-funding-data-for-zimbabwe-showcase") # first resource

## End(Not run)
```

`pull_tag` *Read an HDX tag*

Description

Read an HDX tag from its name or id

Usage

```
pull_tag(
  identifier = NULL,
  vocabulary_id = NULL,
  include_datasets = FALSE,
  configuration = NULL
)
```

Arguments

`identifier`	character the name or id of the tag
`vocabulary_id`	character the id or name of the tag vocabulary that the tag is in - if it is not specified it will assume it is a free tag.
`include_datasets`	logical, include a list of the tag's datasets.
`configuration`	a Configuration object

Value

Tag the tag

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- pull_tag("covid19")
 res

## End(Not run)
```

`pull_user` *Read an HDX user*

Description

Read an HDX user from its name or id

Usage

```
pull_user(
  identifier = NULL,
  include_datasets = FALSE,
  configuration = NULL,
  ...
)
```

Arguments

`identifier`	character user keyword
`include_datasets`	Logical, if TRUE add datasets
`configuration`	a Configuration object
`...`	Extra parameters

Value

User the user

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- pull_user("xxxx")
 res

## End(Not run)
```

pull_vocabulary *Read an HDX Vocabulary*

Description

Read an HDX vocabulary from its name or id

Usage

```
pull_vocabulary(identifier = NULL, configuration = NULL)
```

Arguments

identifier	character the name or id of the vocabulary
configuration	Configuration

Value

The Vocabulary object

Examples

```
## Not run:
# Setting the config to use HDX default server
 set_rhdx_config()
 res <- pull_vocabulary("xxxx")
 res

## End(Not run)
```

read_resource *Read resource*

Description

Read resource

Usage

```
read_resource(
  resource,
  sheet = NULL,
  layer = NULL,
  format = NULL,
  download_folder = NULL,
  simplify_json = TRUE,
  force_download = FALSE,
  quiet_download = TRUE,
  ...
)
```

Arguments

`resource` Resource, an HDX resource

`sheet` Character, the name of the sheet to read if XLS(X) resources. The first sheet is read by default.

`layer` Character, the name of the layer to read if spatial data. The first sheet is read by default.

`format` Character, file format, csv, zipped csv, excel, xlsx, zipped shapefile, etc.

`download_folder` Character, the path of the folder to store the downloaded data

`simplify_json` Logical, if TRUE simplifies nested lists into vectors and data frames for JSON resources

`force_download` Logical, force download if TRUE

`quiet_download` logical, silent download

`...` extra parameters

Value

an `tibble`, a `list`, a `stars` or a `sf` object depending on the type of resource you are reading from HDX

`rhdx` *rhdx*

Description

R client for the Humanitarian Data Exchange platform

Author(s)

`<mail@ahmadoudicko.com>`

rhdx_cache *Caching HDX downloaded files*

Description

Manage cached HDX downloaded files

Usage

```
rhdx_cache_set_dir(path)

rhdx_cache_get_dir()

rhdx_cache_list()

rhdx_cache_delete(file)

rhdx_cache_clear()
```

Arguments

`path`	Character directory to set
`file`	Character, the file to delete

Details

The default cache directory is `~/.cache/R/rhdx_cache`, but you can set your own path using `rhdx_cache_set_dir()`

Value

the cache directory

the cache directory

list of files in the cache

Examples

```
## Not run:
rhdx_cache
## change the default cache directory
tmp <- tempdir()
rhdx_cache_set_dir(tmp)

## print current cache directory
rhdx_cache_get_dir()

## List available files in the current cache directory
```

```
rhdx_cache_list()

l <- rhdx_cache_list()[1] ## get the first file
rhdx_cache_delete(l) ## delete it

rhdx_cache_clear() ## delete all cached files

## End(Not run)
```

`search_datasets` *Search for datasets on HDX*

Description

Search for datasets on HDX

Usage

```
search_datasets(
  query = "*:*",
  filter_query = NULL,
  rows = 10L,
  start = 0L,
  page_size = 1000L,
  configuration = NULL,
  ...
)
```

Arguments

`query` Character Query terms, use solr format and default to ":" (match everything)

`filter_query` Character Filter Query results

`rows` integer; Number of matching records to return. Defaults to 10.

`start` integer; the offset in the complete result for where the set of returned datasets should begin.

`page_size` integer; Size of page to return. Defaults to 1000.

`configuration` Configuration object.

`...` Extra parameters for `package_search` endpoints

Details

Search and find datasets on HDX

Value

A list of HDX datasets

Examples

```
## Not run:
 # Setting the config to use HDX default server
 search_datasets("displaced nigeria", rows = 3L)

## End(Not run)
```

---

search_resources *Search resources*

Description

Search Resources

Usage

```
search_resources(query = "*:*", configuration = NULL, ...)
```

Arguments

query	Character, a query
configuration	a Configuration object
...	extra params

---

search_tags *Search for datasets on HDX*

Description

Search for datasets on HDX

Usage

```
search_tags(
  query = "",
  vocabulary_id = NULL,
  limit = NULL,
  offset = NULL,
  configuration = NULL
)
```

Arguments

`query`	(character) - character to search for
`vocabulary_id`	(character) - the id or name of the tag vocabulary to search in
`limit`	(integer) - the maximum number of tags to return
`offset`	(integer) - when `limit` is given, the offset to start returnings tags from
`configuration`	Configuration object.

Details

Search and find tags on HDX

Value

A list of HDX tags

Examples

```
## Not run:
 # Setting the config to use HDX default server
 search_tags("idps", rows = 3L)

## End(Not run)
```

`set_rhdx_config` *Set rhdx config*

Description

Sets the configuration settings for using rhdx.

Usage

```
set_rhdx_config(
  hdx_site = "prod",
  hdx_key = NULL,
  read_only = TRUE,
  hdx_config = NULL,
  hdx_config_file = NULL,
  configuration = NULL
)

get_rhdx_config()
```

Arguments

`hdx_site`	Character to specify which HDX server you want to use. Default to "prod".
`hdx_key`	Character for the CKAN API key, it is required to push data into HDX
`read_only`	Logical if `FALSE` and hdx_key provided is correct you can push metdata and data to HDX
`hdx_config`	List of HDX configuration
`hdx_config_file`	Character, path of the HDX config file in JSON and YAML format
`configuration`	Configuration object.

Details

Setting up a configuration will help you access from an HDX server

Value

Invisibly returns the rhdx config object

Examples

```
## Not run:
# Setting the config to use HDX default server
set_rhdx_config(hdx_site = "demo")

# You can check your configuration using \code{get_rhdx_config}
config <- get_rhdx_config()
config

## End(Not run)
```

`Tag` *HDX Tag*

Description

HDX Tag

HDX Tag

Details

HDX tag

Super class

`rhdx::HDXObject` -> `Tag`

Public fields

`data` list of tag field element

Methods

Public methods:

- `Tag$new()`
- `Tag$as_list()`
- `Tag$print()`
- `Tag$clone()`

Method `new()`**:** Create a new Tag object

Usage:

```
Tag$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list data with required fields to create a tag object

`configuration` Configuration configuration to use

Returns: a Tag object

Method `as_list()`**:** Tag object to list

Usage:

```
Tag$as_list()
```

Returns: a list with of tag fields element

Method `print()`**:** Print a Tag object

Usage:

```
Tag$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Tag$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Vocabulary` *HDX Vocabulary*

Description

HDX Vocabulary

HDX Vocabulary

Details

HDX Vocabulary

Super class

`rhdx::HDXObject` -> `Vocabulary`

Public fields

`data` list of tag field element

Methods

Public methods:

- `Vocabulary$new()`
- `Vocabulary$as_list()`
- `Vocabulary$authorized_tags_name()`
- `Vocabulary$print()`
- `Vocabulary$clone()`

Method `new()`**:** Create a new Tag object

Usage:

```
Vocabulary$new(initial_data = NULL, configuration = NULL)
```

Arguments:

`initial_data` list data with required fields to create a tag object

`configuration` Configuration configuration to use

Returns: a Tag object

Method `as_list()`**:** Tag object to list

Usage:

```
Vocabulary$as_list()
```

Returns: a list with of tag fields element

Method `authorized_tags_name()`**:** List of accepted tags on HDX

Usage:

```
Vocabulary$authorized_tags_name()
```

Returns: a vector of tags name

Method `print()`**:** Print a Vocabulary object

Usage:

```
Vocabulary$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Vocabulary$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index