

Package: riddle (via r-universe)

October 11, 2026

Title An API wrapper to interact with the UNHCR RIDL Platform

Version 0.0.5

Maintainer Edouard Legoupil <legoupil@unhcr.org>

Description The package wraps functions to work with the RIDL API
ridl.unhcr.org from R.

License MIT + file LICENSE

URL <https://edouard-legoupil.github.io/riddle/>

BugReports <https://github.com/edouard-legoupil/riddle/issues>

Depends R (>= 2.10)

Imports dplyr, glue, here, httr, magrittr, purrr, rlang, rmarkdown,
stringr, tibble, tidyr, tidyselect, unhcrdown

Suggests knitr, testthat

VignetteBuilder knitr

Remotes vidonne/unhcrdown

Config/fusen/version 0.5.2

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-
dev make libharfbuzz-dev libmagick++-dev gsfonts libicu-dev libjpeg-dev libpng-dev libtiff-
dev libuv1-dev libwebp-dev libxml2-dev libssl-dev zlib1g-dev

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

Date/Publication 2024-06-05 14:20:44 UTC

RemoteUrl <https://github.com/Edouard-Legoupil/riddle>

RemoteRef HEAD

RemoteSha 8b767923d83aad6658f8c1240c7eb0e9b48ea22a

Contents

container_list	2
container_show	3
dataset	3
dataset_metadata	5
dataset_tibblify	9
find_child_containers	10
keywords	10
resource	11
resource_fetch	13
resource_metadata	15
resource_tibblify	17
riddle_notebook	18
ridl	19
search	20
summary_report	21
Index	22

container_list	<i>container_list</i>
----------------	-----------------------

Description

Provide a list of all child containers for a specific container

Usage

container_list(parent)

Arguments

parent	name of the container
--------	-----------------------

Details

This function is used to generate a regional dashbaord.. Be carefull - it's an expansive functions at it needs to parse the entire content of the server...

uses https://docs.ckan.org/en/2.9/api/index.html#ckan.logic.action.get.organization_list

Value

catalog of containers

Examples

```
# catalog <- container_list()
# groups_name <- catalog |>
#           dplyr::select(groups_name) |>
#           dplyr::distinct()
```

container_show	<i>container_show</i>
----------------	-----------------------

Description

Get an overview of accessible infos about all containers in RIDL Use <https://docs.ckan.org/en/2.9/api/index.html#ckan.logic.a>

Usage

```
container_show(id)
```

Arguments

id id or name of the container

Value

a dataframe with container metadata

Examples

```
# americasdataset <- container_show( id = "americas-regional-dataset")
```

dataset	<i>Work with RIDL datasets (datasets)</i>
---------	---

Description

Work with RIDL datasets (datasets)

Usage

```
dataset_create(metadata)

dataset_show(id)

dataset_update(id, metadata)

dataset_patch(id, metadata)

dataset_delete(id)
```

Arguments

metadata	Metadata created by <code>dataset_metadata()</code> .
id	The id or name of the dataset.

Details

You must have the necessary permissions to create, edit, or delete datasets.

Note that several fields are required for `dataset_create()` and `dataset_update()` operations to succeed. Consult `dataset_metadata()` for the details.

For `dataset_update()/dataset_patch()` operations, it is recommended to call `dataset_show()`, make the desired changes to the result, and then call `dataset_update()/dataset_patch()` with it.

The difference between the update and patch methods is that the patch will perform an update of the provided parameters, while leaving all other parameters unchanged, whereas the update methods deletes all parameters not explicitly provided in the metadata.

Value

The dataset.

Examples

```
#-----
# test search in prod
Sys.unsetenv("USE_UAT")
# riddle::dataset_show(id = "unhcr-cbi-americas-quarterly-report")
#
# p <- riddle::dataset_show('rms_v4')
# list_of_ressources <- p[["resources"]][[1]]
# list_of_ressources

#-----
# Test create in UAT
Sys.setenv(USE_UAT=1)
m <- riddle::dataset_metadata(title = "Testing Riddle Interface",
                             name = "riddleapitest",
                             notes = "Making an API test",
                             owner_org = "americas", ## be careful- all lower case!!!
                             visibility = "public",
                             geographies = "UNSPECIFIED",
                             external_access_level = "open_access",
                             data_collector = "Motor Trend",
                             keywords = keywords[c("Environment", "Other")],
                             unit_of_measurement = "car",
                             data_collection_technique = "oth",
                             archived = "False")
# ## For the above to work - you need to make sure you have at least editor access
# to the corresponding container - i.e. owner_org = "exercise-container"
```

```

# p <- dataset_create(metadata = m)

# The return value is a representation of the dataset we just created in
# RIDL that you could inspect like any other R object.
# p
## Now deleting this!
# dataset_delete(id = p$id)

#-----
# Test create in prod
Sys.unsetenv("USE_UAT")
# m1 <- riddle::dataset_metadata(title = "Test",
#                                name = "Test",
#                                notes = "The data was extracted from kobo.",
#                                owner_org = "americas-regional-dataset",
#                                visibility = "public",
#                                geographies = "UNSPECIFIED",
#                                external_access_level = "open_access",
#                                data_collector = "UNHCR",
#                                keywords = keywords[c("Environment", "Other")],
#                                unit_of_measurement = "car",
#                                data_collection_technique = "oth",
#                                archived = "False")
# p <- riddle::dataset_create(metadata = m1)

```

dataset_metadata

Convenience function to record dataset metadata

Description

This function create a metadata object used to then interact with the API

Usage

```

dataset_metadata(
  title = NULL,
  name = NULL,
  short_title = NULL,
  notes = NULL,
  tag_string = NULL,
  url = NULL,
  owner_org = NULL,
  geographies = "UNSPECIFIED",
  private = NULL,
  visibility = NULL,
  external_access_level = NULL,
  data_sensitivity = NULL,

```

```

original_id = NULL,
data_collector = NULL,
date_range_start = NULL,
date_range_end = NULL,
keywords = NULL,
unit_of_measurement = NULL,
sampling_procedure = NULL,
operational_purpose_of_data = NULL,
`hxl-ated` = NULL,
process_status = NULL,
identifiability = NULL,
geog_coverage = NULL,
data_collection_technique = NULL,
linked_datasets = NULL,
archived = NULL,
admin_notes = NULL,
sampling_procedure_notes = NULL,
response_rate_notes = NULL,
data_collection_notes = NULL,
weight_notes = NULL,
clean_ops_notes = NULL,
data_accs_notes = NULL,
ddi = NULL,
...
)

```

Arguments

title	Title(*) - Make sure to include: 'Survey name/title', 'Location', 'Country', and 'Year(s)' in the order indicated.
name	URL(*) - The canonical name of the dataset, eg. my-dataset.
short_title	Short title - eg. Short title for the project.
notes	Description(*) - Some useful notes about the data. Please include the number of observations.
tag_string	Tags - eg. economy, mental health, government.
url	Project URL - Website URL associated with this data project (if applicable).
owner_org	Data container(*) - Use the canonical name for the container (i.e. all lower case) for instance "americas" - not "Americas" - in case you are not using the right container you will receive. The id of the container can also be used
geographies	defaults is geographies - pulling from a webservice from geoserver
private	Visibility (Private/Public).
visibility	Internal Access Level(*). Allowed values: restricted (Private), public (Internally Visible).
external_access_level	External access level(*). Allowed values: not_available (Not available), direct_access (Direct access), public_use (Public use), licensed_use (Licensed use), data_enclave (Data enclave), open_access (Open access).

data_sensitivity	Data sensitivity - Apply to both Anonymized and Personally identifiable data. Allowed values: yes (Yes), no (No).
original_id	Original ID - If the dataset already has an ID from the source org, DDI, etc...
data_collector	Data Collector(*) - Which organization owns / collected the data. Multiple values are allowed.
date_range_start	Date collection first date - Use dd/mm/yyyy format.
date_range_end	Date collection last date - Use dd/mm/yyyy format.
keywords	Topic classifications(*) - Tags useful for searching for the datasets. Multiple values are allowed. See keywords
unit_of_measurement	Unit of measurement(*) - Unit of measurement / observation for the dataset.
sampling_procedure	Sampling Procedure. Multiple values are allowed. Allowed values: total_universe_complete Enumeration (Total universe/Complete enumeration), probability_simple_random (Probability: Simple random), probability_systematic_random (Probability: Systematic random), probability_stratified (Probability: Stratified), probability_stratified_proportional (Probability: Stratified: Proportional), probability_stratified_disproportional (Probability: Stratified: Disproportional), probability_cluster (Probability: Cluster), probability_cluster_simple_random (Probability: Cluster: Simple random), probability_cluster_stratified_random (Probability: Cluster: Stratified random), probability_multistage (Probability: Multistage), nonprobability (Non-probability), nonprobability_availability (Non-probability: Availability), nonprobability_purposive (Non-probability: Purposive), nonprobability_quota (Non-probability: Quota), nonprobability_respondentassisted (Non-probability: Respondent-assisted), mixed_probability_nonprobability (Mixed probability and non-probability), other_other (Use if the sampling procedure is known, but not found in the list..).
operational_purpose_of_data	Operational purpose of data - Classification of the type of data contained in the file. Multiple values are allowed. Allowed values: participatory_assessments (Participatory assessments), baseline_household_survey (Baseline Household Survey), rapid_needs_assessment (Rapid Needs Assessment), protection_monitoring (Protection Monitoring), programme_monitoring (Programme monitoring), population_data (Population Data), cartography (Cartography, Infrastructure & GIS).
process_status	Dataset Process Status. Allowed values: raw (Raw-Uncleaned), cleaned (Cleaned Only), anonymized (Cleaned & Anonymized).
identifiability	Identifiability. Allowed values: personally_identifiable (Personally identifiable), anonymized_enclave (Anonymized 1st level: Data Enclave - only removed direct identifiers), anonymized_scientific (Anonymized 2nd level: Scientific Use File (SUF)), anonymized_public (Anonymized 3rd level: Public Use File (PUF)).
geog_coverage	Geographic Coverage - eg. National coverage, or name of the area, etc.

data_collection_technique	Data collection technique(*). Allowed values: nf (Not specified), f2f (Face-to-face interview), capi (Face-to-face interview: Computerised), cami (Face-to-face interview: Mobile), papi (Face-to-face interview: Paper-and-pencil), tri (Telephone interview), eri (E-mail interview), wri (Web-based interview: audio-visual technology enabling the interviewer(s) and interviewee(s) to communicate in real time), easi (Self-administered questionnaire: E-mail), pasi (Self-administered questionnaire: Paper), sasi (Self-administered questionnaire: SMS/MMS), casi (Self-administered questionnaire: Computer-assisted), cawi (Self-administered questionnaire: Web-based), foc (Face-to-face focus group), tfoc (Telephone focus group), obs (Observation), oth (Other).
linked_datasets	Linked Datasets - Links to other RIDL datasets. It supports multiple selections.
archived	Archived(*) - Allows users to indicate if the dataset is archived or active. Allowed values: False (No), True (Yes).
admin_notes	Admin Notes - General. You can use Markdown formatting here.
sampling_procedure_notes	Admin Notes - Sampling Procedure. You can use Markdown formatting here.
response_rate_notes	Admin Notes - Response Rate. You can use Markdown formatting here.
data_collection_notes	Admin Notes - Data Collection. You can use Markdown formatting here.
weight_notes	Admin Notes - Weighting. You can use Markdown formatting here.
clean_ops_notes	Admin Notes - Cleaning. You can use Markdown formatting here.
data_accs_notes	Admin Notes - Access authority. You can use Markdown formatting here.
ddi	DDI.
...	ignored.
'hxl-ated'	HXL-ated. Allowed values: False (No), True (Yes).

Details

All arguments are of type character. Fields `tag_string`, `data_collector`, `keywords`, `sampling_procedure`, and `operational_purpose_of_data` accept vectors of multiple values.

Fields marked with a (*) are required for `dataset_create()` and `dataset_update()` operations.

Value

A list with the provided metadata.

Examples

```
m <- dataset_metadata(title = "Motor Trend Car Road Tests",
                      name = "mtcars",
                      notes = "The data was extracted from the 1974 Motor Trend
```

```

US magazine, and comprises fuel consumption and 10 aspects
of automobile design and performance for 32 automobiles
(1973-74 models).",
owner_org = "americas",
visibility = "public",
geographies = "UNSPECIFIED",
external_access_level = "open_access",
data_collector = "Motor Trend",
keywords = keywords[c("Environment", "Other")],
unit_of_measurement = "car",
data_collection_technique = "oth",
archived = "False")

```

m

dataset_tibblify	<i>dataset_tibblify</i>
------------------	-------------------------

Description

Helper function to package API results as a tibble

Usage

```
dataset_tibblify(x)
```

Arguments

x dataset as a list

Value

dataset

Examples

```

m <- dataset_metadata(title = "Motor Trend Car Road Tests",
  name = "mtcars",
  notes = "The data was extracted from the 1974 Motor Trend
US magazine, and comprises fuel consumption and 10 aspects
of automobile design and performance for 32 automobiles
(1973-74 models).",
  owner_org = "americas", ## becarefull- all lower case!!!
  visibility = "public",
  geographies = "UNSPECIFIED",
  external_access_level = "open_access",
  data_collector = "Motor Trend",
  keywords = keywords[c("Environment", "Other")],
  unit_of_measurement = "car",
  data_collection_technique = "oth",

```

```

        archived = "False")

m1 <- dataset_tibblify(m)
m1

```

find_child_containers *find_child_containers*

Description

Provide a list of all child containers - including nested one - for a specific container

Usage

```
find_child_containers(parent, catalog)
```

Arguments

parent	name of the parent container
catalog	daaframe object with a catalog of container produced by container_list()

Details

Be carefull - it's an expansive functions at it needs to parse the entire content of the server...

Value

vector with all child container

Examples

```

#catalog <- container_list()
# containerAmericas <- find_child_containers(parent = "americas",
#                                           catalog = catalog)

```

keywords	<i>dataset keywords</i>
----------	-------------------------

Description

As extracted from the [dataset schema](#).

Usage

```
keywords
```

Format

A named character vector mapping user-visible labels (the names) to their corresponding codes in the system (the values).

resource	<i>Work with RIDL resources (files)</i>
----------	---

Description

Work with RIDL resources (files)

Usage

```
resource_create(package_id, res_metadata)
resource_update(id, res_metadata)
resource_upload(package_id, res_metadata)
resource_patch(id, res_metadata)
resource_delete(id)
```

Arguments

package_id	The id or name of the dataset to which this resource belongs to.
res_metadata	Metadata created by resource_metadata() .
id	The id or name of the resource.

Details

You must have the necessary permissions to create, edit, or delete datasets and their resources.

Note that several fields are required for `resource_update()`, `resource_create()` and `resource_update()` operations to succeed. Consult [resource_metadata\(\)](#) for the details.

`resource_update()` will check if the resource exists in the dataset. If the resource name does not exist in the dataset, `resource_update()` will create a new resource. If the resource name already exists in the dataset, `resource_update()` will upload the resource and also increase the number in the version.

For `resource_update()/resource_patch()` operations, it is recommended to call `resource_show()`, make the desired changes to the result, and then call `resource_update()/resource_patch()` with it.

The difference between the update and patch methods is that the patch will perform an update of the provided parameters, while leaving all other parameters unchanged, whereas the update methods deletes all parameters not explicitly provided in the metadata.


```

#                               visibility = "public" )

# r <- resource_create(package_id = p$id, res_metadata = new_attachment )
# resource_create(package_id = p$name, res_metadata = new_attachment )
# ## Like before, the return value is a tibble representation of the resource.
# r

# ## Another example with a data resource
# m <- riddle::resource_metadata(type = "data",
#                               url = "mtcars.csv",
#                               upload = httr::upload_file(system.file("extdata/mtcars.csv", package = "readr")),
#                               name = "mtcars.csv",
#                               format = "csv",
#                               file_type = "microdata",
#                               date_range_start = "1973-01-01",
#                               date_range_end = "1973-12-31",
#                               version = "1",
#                               visibility = "public",
#                               process_status = "raw",
#                               identifiability = "anonymized_public")
# r <- resource_create(package_id = p$id,
#                       res_metadata = m )
# ## let's get again the details of the dataset we want to add the resource in..
# r

# ## and now can search for it - checking it is correctly there...
# resource_search("name:mtcarsriddle")

# ## And once we're done experimenting with the API, we should take down our
# ## toy dataset since we don't really need it on RIDL.
# dataset_delete(p$id)

# The return value is a representation of the dataset we just created in
# RIDL that you could inspect like any other R object.
# p
## Now deleting this!
# dataset_delete(id = p$id)

```

resource_fetch

Fetch resource from RIDL

Description

Fetch resource from RIDL

Usage

```
resource_fetch(url, path = tempfile())
```

Arguments

url The URL of the resource to fetch
 path Location to store the resource

Value

Path to the downloaded file

Examples

```
## Example 1: with a direct URL
#-----
# Test search in prod
# Sys.unsetenv("USE_UAT")

# resource_fetch(url = 'https://rid1.unhcr.org/dataset/a60f4b79-8acc-4893-8fb9-d52f94416b19/resource/daa2b9e4-b
# path = tempfile())

## Example 2: Let's try to identify a resource - then fetch it locally and update it back... as from here
# https://github.com/unhcr-americas/darien_gap_human_mobility/blob/main/report.Rmd#L38
# Sys.unsetenv("USE_UAT")
# ## Get the dataset metadata based on its canonical name
# p <- riddle::dataset_show('rms_v4')
# ## Let's get the fifth resource within this dataset
# test_ressources <- p[["resources"]][[1]] |> dplyr::slice(5)
#
# ## Download the resource locally in a file name file..
# resource_fetch(url = test_ressources$url, path = here::here("file"))
# test_ressources$url
# # Rebuild the metadata
# m <- resource_metadata(type = test_ressources$type, #"data",
#                         url = "df_gender_2020.csv",
# upload = httr::upload_file(here::here("file")),
#                         name = test_ressources$name,
# # "Irregular entries by gender in 2022",
#                         format = test_ressources$format, #"csv",
#                         file_type = test_ressources$file_type, #"microdata",
#                         visibility = test_ressources$visibility, # "public",
#                         date_range_start = test_ressources$date_range_start,
# "2022-01-01",
#                         date_range_end = test_ressources$date_range_end, #as.character(floor_date(today('America/Pana
#end day of last month
#                         version = test_ressources$version, # "0",
#                         process_status = test_ressources$process_status,
# "anonymized",
#                         identifiability = test_ressources$identifiability, #"anonymized_public"
# )

#r <- resource_update(id = test_ressources$id, res_metadata = m)
```

resource_metadata	<i>Convenience function to record resource metadata</i>
-------------------	---

Description

This functions create the resource metadata

Usage

```
resource_metadata(
  type = NULL,
  url = NULL,
  name = NULL,
  description = NULL,
  format = NULL,
  file_type = NULL,
  date_range_start = NULL,
  date_range_end = NULL,
  upload = NULL,
  visibility = NULL,
  version = NULL,
  `hxl-ated` = NULL,
  process_status = NULL,
  identifiability = NULL,
  ...
)
```

Arguments

type	Resource type(*) - The kind of file you want to upload. Allowed values: data (Data file), attachment (Additional attachment).
url	Upload - The file name as it will be recorded in the system.
name	Name - eg. January 2011 Gold Prices.
description	Description - Some usefule notes about the data.
format	File format - eg. CSV, XML, or JSON.
file_type	File type(*) - Indicates what is contained in the file. Allowed values: microdata (Microdata), questionnaire (Questionnaire), report (Report), sampling_methodology (Sampling strategy & methodology Description), infographics (Infographics & Dashboard), script (Script), concept note (Concept Note), other (Other).

date_range_start	Data collection first date(*) - Use yyyy-mm-dd format.
date_range_end	Data collection last date(*) - Use yyyy-mm-dd format.
upload	File to upload. Passed using <code>httr::upload_file()</code> .
visibility	should be either
version	Version(*).
process_status	File process status(*) - Indicates the processing stage of the data. 'Raw' means that the data has not been cleaned since collection. 'In process' means that it is being cleaned. 'Final' means that the dataset is final and ready for use in analytical products. Allowed values: raw (Raw-Uncleaned), cleaned (Cleaned Only), anonymized (Cleaned & Anonymized).
identifiability	Identifiability(*) - Indicates if personally identifiable data is contained in the dataset. Allowed values: personally_identifiable (Personally identifiable), anonymized_enclave (Anonymized 1st level: Data Enclave - only removed direct identifiers), anonymized_scientific (Anonymized 2nd level: Scientific Use File (SUF)), anonymized_public (Anonymized 3rd level: Public Use File (PUF)).
...	ignored.
'hxl-ated'	HXL-ated. Allowed values: False (No), True (Yes).

Details

All arguments are of type character.

Fields marked with a (*) are required for `resource_create()` and `resource_update()` operations.

Value

A list with the provided metadata.

Examples

```
#resource_metadata()
m <- riddle::resource_metadata(type = "data",
                              url = "mtcars.csv",
                              name = "mtcars.csv",
                              format = "csv",
                              file_type = "microdata",
                              date_range_start = "1973-01-01",
                              date_range_end = "1973-12-31",
                              version = "1",
                              visibility = "public",
                              process_status = "raw",
                              identifiability = "anonymized_public")
m
```

resource_tibblify	<i>resource_tibblify</i>
-------------------	--------------------------

Description

Helper function to package API results as a tibble

Usage

```
resource_tibblify(x)
```

Arguments

x	list
---	------

Value

list tiblified

Examples

```
m <- riddle::resource_metadata(type = "data",
  url = "mtcars.csv",
  # upload = htr::upload_file(system.file("extdata/mtcars.csv", package = "readr")),
  name = "mtcars.csv",
  format = "csv",
  file_type = "microdata",
  date_range_start = "1973-01-01",
  date_range_end = "1973-12-31",
  version = "1",
  visibility = "public",
  process_status = "raw",
  identifiability = "anonymized_public")

m1 <- riddle::resource_tibblify(m)
```

m1

riddle_notebook	<i>riddle_notebook</i>
-----------------	------------------------

Description

Archive all crunching files in RIDL

Usage

```
riddle_notebook(ridl, datafolder, namethisfile, visibility = "public")
```

Arguments

ridl	ridl container where the resources should be added
datafolder	folder where the data used by the notebook are stored
namethisfile	all files are archived based on the name of notebook you created. The function automatically get the name of the notebook where it is run from, using <code>base::name(rstudioapi::getSourceEditorContext())\$path</code>
visibility	can be "public" per default or set to private for obscure reasons..

Details

RIDL is UNHCR instance of a CKAN server and is accessible for UNHCR staff at <https://ridl.unhcr.org>. It is designed to keep track and document dataset within an organisation.

You conveniently archive there your generated report and save the work you did on a notebook: As you have been working on the data, you want to keep track of it and save your work in a place where it can be useful for other people and available for peer review and quality assessment.

The function saves within the the RIDL container you used to get the data from the following resources:

- the generated report
- the source notebook

The function behavior is the following -

1. Get metadata from the RIDL dataset
2. check if the resources to be uploaded is already shared based on the name
3. if already there update, if not create

The function relies on `# install.packages("pak") # pak::pkg_install("edouard-legoupil/riddle")`

Value

nothing all analysis files are added as a resources

Examples

```
## Time to archive your work once done!!
# used in the RIDL_Notebook markdown template in the package
# if( params$publish == "yes"){
#   namethisfile = basename(rstudioapi::getSourceEditorContext()$path )
#   riddle_notebook(ridl = params$ridl,
#                     datafolder = params$datafolder,
#                     namethisfile = namethisfile ,
#                     visibility = params$visibility ) }
```

ridl	<i>apihelper</i>
------	------------------

Description

Helper function to make **API calls**. Calls includes the 10 following actions:

Usage

```
ridl(action, ..., .encoding = "json", verbose = FALSE)
```

Arguments

action	Operation to execute. See CKAN's API documentation for details.
...	whatever is needed
.encoding	HTTP POST encoding to use - one of json, form, or multipart.
verbose	TRUE FALSE to display info on the console about the API call

Details

On dataset

- "package_create"
- "package_update"
- "package_patch"
- "package_delete"
- "package_search"

On resource

- "resource_create"
- "resource_update"
- "resource_patch"
- "resource_delete"
- "resource_search"

The package works with both the production and UAT instances of RIDL. To use the UAT version, run `Sys.setenv(USE_UAT=1)` before calling any functions from the package. To go back to the production instance, call `Sys.unsetenv("USE_UAT")`.

Value

httr::response object with the result of the call.

Examples

```
# ridl(action = "package_search", as.list("cbi"))
```

search

Searches for datasets and resources satisfying a given criteria.

Description

Searches for datasets and resources satisfying a given criteria.

Usage

```
dataset_search(q = NULL, rows = NULL, start = NULL)
```

```
resource_search(query = NULL, rows = NULL, start = NULL)
```

Arguments

q, query	The search query.
rows	The maximum number of matching rows (datasets) to return. (optional, default: 10, upper limit: 1000)
start	The offset in the complete result for where the set of returned datasets should begin.

Value

A tibble with the search results.
tibble with list of related resource.

Examples

```
#-----
# Test search in prod
# Sys.unsetenv("USE_UAT")
# searching <- "cbi"
# p <- dataset_search(q = searching, rows = 30)
# p
```

```
#-----
# Test create in UAT
Sys.setenv(USE_UAT=1)
# p2 <- dataset_search(q = "testedouard2")
```

summary_report	<i>Generate a RIDL factsheet</i>
----------------	----------------------------------

Description

Generate a RIDL factsheet

Usage

```
summary_report(container = "Americas")
```

Arguments

container list of container to generate the factsheet to generate

Examples

```
# summary_report(year = 2022,  
#                               region = "Americas")
```

Index

* **datasets**

keywords, [10](#)

container_list, [2](#)

container_show, [3](#)

dataset, [3](#)

dataset_create (dataset), [3](#)

dataset_create(), [8](#)

dataset_delete (dataset), [3](#)

dataset_metadata, [5](#)

dataset_metadata(), [4](#)

dataset_patch (dataset), [3](#)

dataset_search (search), [20](#)

dataset_show (dataset), [3](#)

dataset_tibblify, [9](#)

dataset_update (dataset), [3](#)

dataset_update(), [8](#)

find_child_containers, [10](#)

keywords, [7](#), [10](#)

resource, [11](#)

resource_create (resource), [11](#)

resource_create(), [16](#)

resource_delete (resource), [11](#)

resource_fetch, [13](#)

resource_metadata, [15](#)

resource_metadata(), [11](#)

resource_patch (resource), [11](#)

resource_search (search), [20](#)

resource_tibblify, [17](#)

resource_update (resource), [11](#)

resource_update(), [16](#)

resource_upload (resource), [11](#)

riddle_notebook, [18](#)

ridl, [19](#)

search, [20](#)

summary_report, [21](#)