Package ‘ecmwfr’

January 19, 2023

Title Interface to ‘ECMWF’ and ‘CDS’ Data Web Services

Version 1.5.0

Description Programmatic interface to the European Centre for Medium-Range Weather Forecasts dataset web services (ECMWF; <https://www.ecmwf.int/>) and Copernicus’s Climate Data Store (CDS; <https://cds.climate.copernicus.eu>). Allows for easy downloads of weather forecasts and climate reanalysis data in R.

URL https://github.com/bluegreen-labs/ecmwfr

BugReports https://github.com/bluegreen-labs/ecmwfr/issues

Depends R (>= 3.6)

Imports httr, keyring, memoise, getPass, curl, R6, uuid

License AGPL-3

ByteCompile true

RoxygenNote 7.2.1

Suggests rmarkdown, covr, testthat, terra, maps, ncdf4, knitr, rlang, rstudioapi, jsonlite

VignetteBuilder knitr

NeedsCompilation no

Author Koen Hufkens [aut, cre] (<https://orcid.org/0000-0002-5070-8109>), Reto Stauffer [ctb] (<https://orcid.org/0000-0002-3798-5507>), Elio Campitelli [ctb] (<https://orcid.org/0000-0002-7742-9230>), BlueGreen Labs [cph, fnd]

Maintainer Koen Hufkens <koen.hufkens@gmail.com>

Repository CRAN

Date/Publication 2023-01-19 13:00:02 UTC

R topics documented:

  wf_archetype .......................................................... 2
  wf_check_request ..................................................... 3
Description

Creates a universal MARS / CDS formatting function, in ways similar to `wf_modify_request()` but the added advantage that you could code for the use of dynamic changes in the parameters provided to the resulting custom function.

Usage

`wf_archetype(request, dynamic_fields)`

Arguments

- `request` a MARS or CDS request as an R list object.
- `dynamic_fields` character vector of fields that could be changed.

Details

Contrary to a simple replacement as in `wf_modify_request()` the generated functions are considered custom user written. Given the potential for complex formulations and formatting commands NO SUPPORT for the resulting functions can be provided. Only the generation of a valid function will be guaranteed and tested for.

Value

a function that takes ‘dynamic_fields’ as arguments and returns a request as an R list object.

Examples

```r
## Not run:
# format an archetype function
ERAI <- wf_archetype(
  request = list(stream = "oper",
                 levtype = "sfc",
                 param = "165.128/166.128/167.128",
```
dataset = "interim",
step = "0",
grid = "0.75/0.75",
time = "00/06/12/18",
date = "2014-07-01/to/2014-07-31",
type = "an",
class = "ei",
area = "73.5/-27/33/45",
format = "netcdf",
target = "tmp.nc"),
dynamic_fields = c("date", "time")
)

# print output of the function with below parameters
str(ERA_interim("20100101", 3, 200))

## End(Not run)

---

**Description**

Check the validity of a data request, and login credentials.

**Usage**

```r
wf_check_request(user, request)
```

**Arguments**

- **user**: user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by `wf_set_key`
- **request**: nested list with query parameters following the layout as specified on the ECMWF API page

**Value**

a data frame with the determined service and url service endpoint

**Author(s)**

Koen Hufkens

**See Also**

`wf_set_key`, `wf_transfer`, `wf_request`
Description

Returns a list of datasets

Usage

`wf_datasets(user, service = "webapi", simplify = TRUE)`

Arguments

- `user` user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by `wf_set_key`
- `service` which service to use, one of webapi, cds or ads (default = webapi)
- `simplify` simplify the output, logical (default = TRUE)

Value

returns a nested list or data frame with the ECMWF datasets

Author(s)

Koen Hufkens

See Also

`wf_set_key wf_transfer wf_request`

Examples

```r
## Not run:
# set key
wf_set_key(email = "test@mail.com", key = "123")

# get a list of services
wf_services("test@mail.com")

# get a list of datasets
wf_datasets("test@mail.com")

## End(Not run)
```
Description

Deletes a staged download from the queue

Usage

```r
wf_delete(url, user, service = "webapi", verbose = TRUE)
```

Arguments

- `url`: url to query
- `user`: user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by `wf_set_key`
- `service`: which service to use, one of `webapi`, `cds` or `ads` (default = `webapi`)
- `verbose`: show feedback on processing

Author(s)

Koen Hufkens

See Also

`wf_set_key` `wf_transfer` `wf_request`

Examples

```r
## Not run:
# set key
wf_set_key(email = "test@mail.com", key = "123")

# get key
wf_get_key(email = "test@mail.com")

## End(Not run)
```
**wf_get_key**  
*Get secret ECMWF / CDS token*

Description

Returns you token set by **wf_set_key**

Usage

```
wf_get_key(user, service = "webapi")
```

Arguments

- **user**: user (email address) used to sign up for the ECMWF data service
- **service**: which service to use, one of webapi, cds or ads (default = webapi)

Value

the key set using **wf_set_key** saved in the keychain

Author(s)

Koen Kufkens

See Also

**wf_set_key**

Examples

```
## Not run:
# set key
wf_set_key(user = "test@mail.com", key = "123")

# get key
wf_get_key(user = "test@mail.com")

## End(Not run)
```
Renders product lists for a given dataset and data service

Description

Shows and returns detailed product information about a specific data set (see `wf_datasets`).

Usage

`wf_product_info(dataset, user, service = "webapi", simplify = TRUE)`

Arguments

- `dataset` character, name of the data set for which the product information should be loaded.
- `user` string, user ID used to sign up for the CDS / ADS data service, used to retrieve the token set by `wf_set_key`.
- `service` which service to use, one of `webapi`, `cds` or `ads` (default = `webapi`)
- `simplify` boolean, default TRUE. If TRUE the description will be returned as tidy data instead of a nested list.

Value

Downloads a tidy data frame with product descriptions from CDS. If `simplify = FALSE` a list with product details will be returned.

Author(s)

Reto Stauffer, Koen Hufkens

See Also

`wf_datasets`.

Examples

```r
## Not run:
# Open description in browser
wf_product_info(NULL, "reanalysis-era5-single-levels")

# Return information
info <- wf_product_info(NULL, 
  "reanalysis-era5-single-levels", show = FALSE)
names(info)

## End(Not run)
```
wf_request  ECMWF data request and download

Description
Stage a data request, and optionally download the data to disk. Alternatively you can only stage requests, logging the request URLs to submit download queries later on using `wf_transfer`. Note that the function will do some basic checks on the request input to identify possible problems.

Usage
```
wf_request(
  request,
  user,
  transfer = TRUE,
  path = tempdir(),
  time_out = 3600,
  job_name,
  verbose = TRUE
)
```

```
wf_request_batch(
  request_list,
  workers = 2,
  user,
  path = tempdir(),
  time_out = 3600,
  total_timeout = length(request_list) * time_out/workers
)
```

Arguments
- **request**: nested list with query parameters following the layout as specified on the ECMWF APIs page
- **user**: user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by `wf_set_key`
- **transfer**: logical, download data TRUE or FALSE (default = TRUE)
- **path**: path were to store the downloaded data
- **time_out**: how long to wait on a download to start (default = 3*3600 seconds).
- **job_name**: optional name to use as an RStudio job and as output variable name. It has to be a syntactically valid name.
- **verbose**: show feedback on processing
- **request_list**: a list of requests that will be processed in parallel.
- **workers**: maximum number of simultaneous request that will be submitted to the service. Most ECMWF services are limited to 20 concurrent requests (default = 2).
- **total_timeout**: overall timeout limit for all the requests in seconds.
Details

Two sorts of requests are accepted, a simple data request based upon the available data in the (raw) CDS repository, and a workflow request which forwards an anonymous python function to the CDS servers and returns its results.

The latter advanced use case is non-trivial, as both python and R code is required. However, it allows you to offload costly data operations / aggregation to the ECMWF servers, therefore limiting the amount of data that needs to be transferred.

A detailed summary of the use of the python API underpinning the CDS Toolbox (Editor) these operations is beyond the scope of this package. We refer to the [CDS Toolbox manual](https://cds.climate.copernicus.eu/toolbox/doc/api.html), and the small example included in the [vignettes](https://bluegreen-labs.github.io/ecmwfr/articles/cds_workflow_vignette.html).

Value

the path of the downloaded (requested file) or the an R6 object with download/transfer information

Author(s)

Koen Hufkens

See Also

`wf_set_key` `wf_transfer`

Examples

```r
## Not run:
# set key
wf_set_key(user = "test@mail.com", key = "123")

request <- list(stream = "oper",
    levtype = "sfc",
    param = "167.128",
    dataset = "interim",
    step = "0",
    grid = "0.75/0.75",
    time = "00",
    date = "2014-07-01/to/2014-07-02",
    type = "an",
    class = "ei",
    area = "50/10/51/11",
    format = "netcdf",
    target = "tmp.nc")

# demo query
wf_request(request = request, user = "test@mail.com")

# Run as an RStudio Job. When finished, will create a
# variable named "test" in your environment with the path to
# the downloaded file.
```
wf_services

Description

Returns a list of services

Usage

wf_services(user, simplify = TRUE)

Arguments

user user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by wf_set_key
simplify simplify the output, logical (default = TRUE)

Value

returns a nested list or data frame with the ECMWF services

See Also

wf_set_key wf_transfer wf_request

Examples

## Not run:
# set key
wf_set_key(user = "test@mail.com", key = "123")

# get a list of services
wf_services("test@mail.com")

# get a list of datasets
wf_services("test@mail.com")

## End(Not run)
**wf_set_key**  
*Set secret ECMWF token*

**Description**

Saves the token to your local keychain under a service called "ecmwfr".

**Usage**

```
wf_set_key(user, key, service)
```

**Arguments**

- **user**: user (email address) used to sign up for the ECMWF data service
- **key**: token provided by ECMWF
- **service**: which service to use, one of webapi, cds or ads

**Details**

In systems without keychain management set the option keyring_backend to ‘file’ (i.e. `options(keyringBackend = "file")`) in order to write the keychain entry to an encrypted file. This mostly pertains to headless Linux systems. The keychain files can be found in `~/.config/r-keyring`.

**Value**

It invisibly returns the user.

**Author(s)**

Koen Hufkens

**See Also**

`wf_get_key`

**Examples**

```r
## Not run:
# set key
wf_set_key(user = "test@mail.com", key = "123")

# get key
wf_get_key(user = "test@mail.com")

# leave user and key empty to open a browser window to the service's website
# and type the key interactively
wf_get_key()
```
### Description

Returns the contents of the requested url as a NetCDF file downloaded to disk or the current status of the requested transfer.

### Usage

```r
wf_transfer(
  url,   # R6 wf_request) query output
  user,  # user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by wf_set_key.
  service = "webapi",  # which service to use, one of webapi, cds or ads (default = webapi)
  path = tempdir(),    # path were to store the downloaded data
  filename = tempfile("ecmwf", tmpdir = ""),  # filename to use for the downloaded data
  verbose = TRUE       # show feedback on data transfers
)
```

### Arguments

- `url`: R6 `wf_request) query output
- `user`: user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by `wf_set_key`
- `service`: which service to use, one of `webapi`, `cds` or `ads` (default = `webapi`)
- `path`: path were to store the downloaded data
- `filename`: filename to use for the downloaded data
- `verbose`: show feedback on data transfers

### Value

A netCDF of data on disk as specified by a `wf_request`.

### Author(s)

Koen Hufkens

### See Also

`wf_set_key`, `wf_request`
Examples

## Not run:
# set key
wf_set_key(user = "test@mail.com", key = "123")

# request data and grab url and try a transfer
r <- wf_request(request, "test@email.com", transfer = FALSE)

# check transfer, will download if available
wf_transfer(r$get_url(), "test@email.com")

## End(Not run)

---

**wf_user_info**  
ECMWF WebAPI user info query

### Description

Returns user info for the ECMWF WebAPI

### Usage

`wf_user_info(user)`

### Arguments

- `user`  
  user (email address) used to sign up for the ECMWF data service, used to retrieve the token set by `wf_set_key`

### Value

returns a data frame with user info

### See Also

`wf_set_key` `wf_services` `wf_datasets`

### Examples

## Not run:
# set key
wf_set_key(user = "test@mail.com", key = "123")

# get user info
wf_user_info("test@mail.com")

## End(Not run)
Index

wf_archetype, 2
wf_check_request, 3
wf_datasets, 4, 7, 13
wf_delete, 5
wf_get_key, 6, 11
wf_product_info, 7
wf_request, 3–5, 8, 10, 12
wf_request_batch (wf_request), 8
wf_services, 10, 13
wf_set_key, 3–10, 11, 12, 13
wf_transfer, 3–5, 8–10, 12
wf_user_info, 13