Package ‘ecmwfr’

February 14, 2020

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

Version  1.2.2

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/khufkens/ecmwfr

BugReports  https://github.com/khufkens/ecmwfr/issues

Depends  R (>= 3.4)

Imports  httr, keyring, memoise, getPass, curl

License  AGPL-3

LazyData  true

ByteCompile  true

RoxygenNote  7.0.2

Suggests  rmarkdown, covr, testthat, raster, 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>)

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

Repository  CRAN

Date/Publication  2020-02-13 23:40:02 UTC
**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**

```r
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)
```

---

**wf_check_request**

**check ECMWF / CDS data requests**

**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 Kufkens

See Also

wf_set_key, wf_transfer, wf_request

---

wf_datasets  ECMWF dataset list

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 service to use ecmwf webapi or cds (default = "webapi")
simplify simplify the output, logical (default = TRUE)

Value

returns a nested list or data frame with the ECMWF datasets

Author(s)
Koen Kufkens

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_delete**

```r
wf_datasets("test@mail.com")
```

## End(Not run)

---

**wf_delete**  

*ECMWF delete request*

### 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`  
  character, one of `ecmwf` or `cds` depending on the data set to be deleted.

- `verbose`  
  show feedback on processing

### Author(s)

Koen Kufkens

### 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` service associated with credentials ("webapi" or "cds")

**Value**

the key set using `wf_set_key` saved in the keychain

**Author(s)**

Koen Kufkens

**See Also**

`wf_set_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")

## End(Not run)
```
wf_product_info

Renders product lists for a given dataset and data service

Description

Shows and returns detailed product information about a specific data set (see \texttt{wf_datasets}).

Usage

\texttt{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 data service, used to retrieve the token set by \texttt{wf_set_key}.
service which service to use, one of \texttt{webapi} or \texttt{cds}
simplify boolean, default \texttt{TRUE}. If \texttt{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 \texttt{simplify} = \texttt{FALSE} a list with product details will be returned.

Author(s)

Reto Stauffer, Koen Hufkens

See Also

\texttt{wf_datasets}.

Examples

## 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)
**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**

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

**Arguments**

- `request`: nested list with query parameters following the layout as specified on the ECMWF API 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

**Value**

a download query staging url or (invisible) filename of the file on your local disc

**Author(s)**

Koen Kufkens

**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_request(request = request, user = "test@mail.com", job_name = "test")

## End(Not run)
```

### wf_services

**ECMWF services list**

**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**

```r
## 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 "ecmwf".

**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`: service associated with credentials ("webapi" or "cds")

**Value**

It invisibly returns the user.

**Author(s)**

Koen Kufkens

**See Also**

`wf_get_key`
**wf_transfer**

**ECMWF data transfer function**

## 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()
```

## End(Not run)

**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,
  user,
  service = "webapi",
  path = tempdir(),
  filename = tempfile("ecmwfr_"),
  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 or cds
- `path`: path 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 Kufkens

See Also

`wf_set_key` `wf_request`

Examples

```r
## 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$href, "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** (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

```r
## 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, 12
wf_delete, 5
wf_get_key, 6, 10
wf_product_info, 7
wf_request, 4, 5, 8, 10, 12
wf_services, 9, 12
wf_set_key, 3–10, 10, 11, 12
wf_transfer, 4, 5, 8, 10, 11
wf_user_info, 12