

# Package: earthdatalogin (via r-universe)

June 13, 2026

**Title** NASA 'EarthData' Access Utilities

**Version** 0.0.4

**Description** Providing easy, portable access to NASA 'EarthData' products through the use of bearer tokens. Much of NASA's public data catalogs hosted and maintained by its 12 Distributed Active Archive Centers ('DAACs') are now made available on the Amazon Web Services 'S3' storage. However, accessing this data through the standard 'S3' API is restricted to only to compute resources running inside 'us-west-2' Data Center in Portland, Oregon, which allows NASA to avoid being charged data egress rates. This package provides public access to the data from any networked device by using the 'EarthData' login application programming interface (API), <https://www.earthdata.nasa.gov/data/earthdata-login>, providing convenient authentication and access to cloud-hosted NASA 'EarthData' products. This makes access to a wide range of earth observation data from any location straight forward and compatible with R packages that are widely used with cloud native earth observation data (such as 'terra', 'sf', etc.)

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 4.1.0)

**Imports** httr, openssl, purrr, utils, base64enc, httr2, jsonlite

**Suggests** knitr, rmarkdown, terra (>= 1.7.39), rsconnect, testthat (>= 3.0.0), curl, sf, fs, readr, spelling, gdalcubes, rstac

**URL** <https://boettiger-lab.github.io/earthdatalogin/>,  
<https://github.com/boettiger-lab/earthdatalogin>

**BugReports** <https://github.com/boettiger-lab/earthdatalogin/issues>

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Language** en-US

**NeedsCompilation** no

**Author** Carl Boettiger [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1642-628X>), Luis López [aut] (ORCID: <https://orcid.org/0000-0003-4896-3263>), Yuvi Panda [aut], Bri Lind [aut] (ORCID: <https://orcid.org/0000-0002-5306-9963>), Andy Teucher [ctb] (ORCID: <https://orcid.org/0000-0002-7840-692X>), Openscapes [fnd]

**Maintainer** Carl Boettiger <cboettig@gmail.com>

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-06-12 22:10:02 UTC

**RemoteUrl** <https://github.com/cran/earthdatalogin>

**RemoteRef** HEAD

**RemoteSha** e7a8804cc7331d0f48459a701f4d81104a25c1f9

## Contents

collections_fetch . . . . .	3
edl_as_s3 . . . . .	3
edl_download . . . . .	4
edl_extract_urls . . . . .	5
edl_netrc . . . . .	6
edl_revoke_token . . . . .	7
edl_s3_token . . . . .	8
edl_search . . . . .	9
edl_set_token . . . . .	11
edl_stac_urls . . . . .	12
edl_unset_netrc . . . . .	12
edl_unset_s3 . . . . .	13
edl_unset_token . . . . .	14
gdal_cloud_config . . . . .	14
gdal_cloud_unconfig . . . . .	15
get_nasa_stac_url . . . . .	16
list_nasa_stacs . . . . .	16
lpdacc_example_url . . . . .	17
with_gdalcubes . . . . .	18

**Index**

**19**

---

collections_fetch	<i>Fetch all collections from an <code>rstac::collections()</code> query</i>
-------------------	--

---

### Description

By default, NASA STAC catalogue collections queries only return 10 collections at a time. This function will page through the collections and return them all.

### Usage

```
collections_fetch(collections, ...)
```

### Arguments

collections	an object of class <code>doc_collections</code> as returned by calling <code>rstac::get_request()</code> on an <code>rstac::collections()</code> query
...	Optional arguments passed on to <code>httr::GET()</code>

### Value

A `doc_collections` object with all of the collections from the catalogue specified in the `rstac::collections()` query.

### Examples

```
rstac::stac("https://cmr.earthdata.nasa.gov/stac/LPCLLOUD") |>  
  rstac::collections() |>  
  rstac::get_request() |>  
  collections_fetch()
```

---

edl_as_s3	<i>Replace https URLs with S3 URIs</i>
-----------	--

---

### Description

Replace https URLs with S3 URIs

### Usage

```
edl_as_s3(href, prefix = "s3://")
```

**Arguments**

href	a https URL from an EarthData Cloud address
prefix	the preferred s3 prefix, e.g. s3:// (understood by gdalcubes), or /vsis3, for terra/stars/sf or other GDAL-based interfaces.

**Value**

a URI that strips basename and protocol and appends prefix

**Examples**

```
href <- lpdacc_example_url()
edl_as_s3(href)
```

---

edl_download	<i>download assets from earthdata over https using bearer tokens</i>
--------------	--

---

**Description**

NOTE: This should be used primarily as a fallback mechanism! EarthData Cloud resources are often best accessed directly over HTTPS without download. This allows subsets to be extracted instead of downloading unnecessary bits. Unfortunately, certain formats do not support such HTTP-based range requests (e.g. HDF4), and require the asset is downloaded to a local POSIX filesystem first.

**Usage**

```
edl_download(
  href,
  dest = basename(href),
  auth = "netrc",
  method = "httr",
  username = default("user"),
  password = default("password"),
  netrc_path = edl_netrc_path(),
  cookie_path = edl_cookie_path(),
  quiet = TRUE,
  ...
)
```

**Arguments**

href	the https URL of the asset
dest	local destination
auth	the authentication method ("token" for Bearer tokens or "netrc" for netrc.)
method	The download method, either "httr" or "curl".

username	EarthData Login User
password	EarthData Login Password
netrc_path	Path to the .netrc file to be created. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
cookie_path	Path to the file where cookies will be stored. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
quiet	logical default TRUE. Show progress in download?
...	additional arguments to <code>download.file()</code> , e.g. <code>quiet = TRUE</code> .

**Value**

the dest path, invisibly

**Examples**

```
href <- lpdacc_example_url()
edl_download(href)
```

---

edl\_extract\_urls      *Extract data URLs from edl\_search*

---

**Description**

**NOTE** this function uses heuristic rules to extract data from `edl_search()`. Users are strongly encouraged to rely on STAC searches instead.

**Usage**

```
edl_extract_urls(items)
```

**Arguments**

items                  the content object from `edl_search`

**Value**

a character vector of URLs

**Examples**

```
items <- edl_search(short_name = "MUR-JPL-L4-GLOB-v4.1",
                    temporal = c("2020-01-01", "2021-12-31"),
                    parse_urls = FALSE)

urls <- edl_extract_urls(items)
```

---

`edl_netrc`*Set up Earthdata Login (EDL) credentials using a .netrc file*

---

## Description

This function creates a .netrc file with Earthdata Login (EDL) credentials (username and password) and sets the necessary environment variables for GDAL to use the .netrc file.

## Usage

```
edl_netrc(  
  username = default("user"),  
  password = default("password"),  
  netrc_path = edl_netrc_path(),  
  cookie_path = edl_cookie_path(),  
  cloud_config = TRUE  
)
```

## Arguments

<code>username</code>	EarthData Login User
<code>password</code>	EarthData Login Password
<code>netrc_path</code>	Path to the .netrc file to be created. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
<code>cookie_path</code>	Path to the file where cookies will be stored. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
<code>cloud_config</code>	set <code>gdal_cloud_config()</code> env vars as well? logical, default TRUE.

## Details

The function sets the environment variables `GDAL_HTTP_NETRC` and `GDAL_HTTP_NETRC_FILE` to enable GDAL to use the .netrc file for EDL authentication. `GDAL_HTTP_COOKIEFILE` and `GDAL_HTTP_COOKIEJAR` are also set to allow the authentication to store and read access cookies.

Additionally, it manages the creation of a symbolic link to the .netrc file if GDAL version is less than 3.7.0 (and thus does not support `GDAL_HTTP_NETRC_FILE` location).

## Value

TRUE invisibly if successful

## Examples

```
edl_netrc()
url <- lpdacc_example_url()
terra::rast(url, vsi=TRUE)
```

---

edl_revoke_token	<i>Revoke an EarthData token</i>
------------------	----------------------------------

---

## Description

Users can only have at most 2 active tokens at any time. You don't need to keep track of a token since earthdatalogin can retrieve your tokens with your user name and password. However, should you want to revoke a token, you can do so with this function.

## Usage

```
edl_revoke_token(
  username = default("user"),
  password = default("password"),
  token_number = 1
)
```

## Arguments

username	EarthData Login User
password	EarthData Login Password
token_number	Which token (1 or 2)

## Value

API response (invisibly)

## Examples

```
edl_revoke_token()
```

---

`edl_s3_token`*NASA Earthdata S3 Credentials Authentication*

---

### Description

Authenticates with NASA Earthdata to obtain temporary S3 credentials for accessing NASA Earth observation data stored in S3 buckets. The function performs an OAuth-like authentication flow and automatically sets the required AWS environment variables.

### Usage

```
edl_s3_token(  
  daac = "https://data.lpdaac.earthdatacloud.nasa.gov",  
  username = default("user"),  
  password = default("password"),  
  prompt_for_netrc = interactive()  
)
```

### Arguments

<code>daac</code>	Character string. The base URL for the DAAC (Data Archive Access Center). Defaults to "https://data.lpdaac.earthdatacloud.nasa.gov".
<code>username</code>	Character string. EarthDataLogin username. Defaults to the value returned by <code>default("user")</code> .
<code>password</code>	Character string. EarthDataLogin password. Defaults to the value returned by <code>default("password")</code> .
<code>prompt_for_netrc</code>	Logical. Often netrc is preferable, so this function will by default prompt the user to switch. Set to FALSE to silence this. Defaults to <code>interactive()</code> .

### Details

Note that these S3 credentials will only work:

- On AWS instance in the us-west-2 region
- Only for one hour before they expire
- Only on the DAAC requested

Please consider using `edl_netrc()` to avoid these limitations

This function performs a multi-step authentication process:

1. Requests authorization URL from the credentials endpoint
2. Posts credentials to get a redirect URL
3. Follows redirect to set authentication cookies
4. Makes final request with cookies to obtain S3 credentials

The function automatically sets the following environment variables:

- AWS\_ACCESS\_KEY\_ID
- AWS\_SECRET\_ACCESS\_KEY
- AWS\_SESSION\_TOKEN

### Value

Invisibly returns a list containing the S3 credentials:

accessKeyId	AWS access key ID
secretAccessKey	AWS secret access key
sessionToken	AWS session token
expiration	Token expiration time

### Examples

```
edl_s3_token()
```

---

edl\_search

*Search for data products using the EarthData API*

---

### Description

**NOTE:** Use as a fallback method only! Users are strongly encouraged to rely on the STAC endpoints for NASA EarthData, as shown in the package vignettes. STAC is a widely used metadata standard by both NASA and many other providers, and can be searched using the feature-rich `rstac` package. STAC return items can be more easily parsed as well.

### Usage

```
edl_search(
  short_name = NULL,
  version = NULL,
  doi = NULL,
  daac = NULL,
  provider = NULL,
  temporal = NULL,
  bounding_box = NULL,
  page_size = 2000,
  recurse = TRUE,
  parse_results = TRUE,
  username = default("user"),
  password = default("password"),
  netrc_path = edl_netrc_path(),
```

```

    cookie_path = edl_cookie_path(),
    ...
)

```

### Arguments

short_name	dataset short name e.g. ATL08
version	dataset version
doi	DOI for a dataset
daac	NSIDC or PODAAC
provider	particular to each DAAC, e.g. POCLLOUD, LPDAAC etc.
temporal	c("yyyy-mm-dd", "yyyy-mm-dd")
bounding_box	c(lower_left_lon, lower_left_lat, upper_right_lon, upper_right_lat)
page_size	maximum number of results to return per query.
recurse	If a query returns more than page_size results, should we make recursive calls to return all results?
parse_results	logical, default TRUE. Calls <code>edl_extract_urls()</code> to determine url links to data objects. Set to FALSE to return the full API response object, but be wary of large object sizes when search returns many results.
username	EarthData Login User
password	EarthData Login Password
netrc_path	Path to the .netrc file to be created. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
cookie_path	Path to the file where cookies will be stored. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
...	additional query parameters

### Value

A character vector of data URLs matching the search criteria, if `parse_results = TRUE` (default). Otherwise, returns a response object of the returned search information if `parse_results = FALSE`.

### Examples

```

items <- edl_search(short_name = "MUR-JPL-L4-GLOB-v4.1",
                  temporal = c("2002-01-01", "2021-12-31"),
                  recurse = TRUE,
                  parse_urls = TRUE)

urls <- edl_extract_urls(items)

```

---

edl\_set\_token

*Get or set an earthdata login token*


---

### Description

This function will ping the EarthData API for any available tokens. If a token is not found, it will request one. You may only have two active tokens at any given time. Use `edl_revoke_token` to remove unwanted tokens. By default, the function will also set an environmental variable for the active R session to store the token. This allows popular R packages which use `gdal` to immediately authenticate any http addresses to NASA EarthData assets.

### Usage

```
edl_set_token(
  username = default("user"),
  password = default("password"),
  token_number = 1,
  set_env_var = TRUE,
  format = c("token", "header", "file"),
  prompt_for_netrc = interactive()
)
```

### Arguments

<code>username</code>	EarthData Login User
<code>password</code>	EarthData Login Password
<code>token_number</code>	Which token (1 or 2)
<code>set_env_var</code>	Should we set the <code>GDAL_HTTP_HEADER_FILE</code> environmental variable? logical, default TRUE.
<code>format</code>	One of "token", "header" or "file." "header" adds the prefix used by http headers to the return string. "file" returns
<code>prompt_for_netrc</code>	Often netrc is preferable, so this function will by default prompt the user to switch. Set to FALSE to silence this.

### Details

**IMPORTANT:** it is necessary to unset this token using `edl_unset_token()` before trying to access HTTP resources that are not part of EarthData, as setting this token will cause those calls to fail! OR simply use `edl_netrc()` to authenticate without facing this issue.

**NOTE:** Because `GDAL >= 3.6.1` is required to recognize the `GDAL_HTTP_HEADERS`, but all versions recognize `GDAL_HTTP_HEADER_FILE`. So we set the Bearer token in a temporary file and provide this path as `GDAL_HTTP_HEADER_FILE` to improve compatibility with older versions.

**Value**

A text string containing only the token (format=token), or a token with the header prefix included, Authorization: Bearer <token>

**Examples**

```
edl_set_token()
edl_unset_token()
```

---

edl_stac_urls	<i>Helper function for extracting URLs from STAC</i>
---------------	--

---

**Description**

Helper function for extracting URLs from STAC

**Usage**

```
edl_stac_urls(items, assets = "data")
```

**Arguments**

items	an items list from rstac
assets	name(s) of assets to extract

**Value**

a vector of hrefs for all discovered assets.

---

edl_unset_netrc	<i>edl_unset_netrc</i>
-----------------	------------------------

---

**Description**

Unsets environmental variables set by edl\_netrc() and removes configuration files set by [edl\\_netrc\(\)](#).

**Usage**

```
edl_unset_netrc(
  netrc_path = edl_netrc_path(),
  cookie_path = edl_cookie_path(),
  cloud_config = TRUE
)
```

**Arguments**

netrc_path	Path to the .netrc file to be created. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
cookie_path	Path to the file where cookies will be stored. Defaults to the appropriate R package configuration location given by <code>tools::R_user_dir()</code> .
cloud_config	set <code>gdal_cloud_config()</code> env vars as well? logical, default TRUE.

**Details**

Note that this function should rarely be necessary, as unlike bearer token-based auth, netrc is mapped by domain name and will not interfere with access to non-earthdata-based URLs. It may still be necessary to deactivate in order to use one of the other earthdatalogin authentication methods.

To unset environmental variables without removing files, set that file path argument to "" (see examples)

Note that GDAL\_HTTP\_NETRC defaults to YES.

**Value**

invisible TRUE, if successful (even if no env is set.)

**Examples**

```
edl_unset_netrc()

# unset environmental variables only
edl_unset_netrc("", "")
```

---

edl\_unset\_s3

*Unset AWS S3 Environment Variables*


---

**Description**

The function uses `Sys.unsetenv()` to remove the specified environment variables.

**Usage**

```
edl_unset_s3()
```

**Details**

This function unsets the AWS S3-related environment variables: `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN`.

**See Also**

[Sys.unsetenv](#)

**Examples**

```
edl_unset_s3()
```

---

edl_unset_token	<i>unset token</i>
-----------------	--------------------

---

**Description**

External sources that don't need the token may error if token is set. Call `edl_unset_token` before accessing non-EarthData URLs.

**Usage**

```
edl_unset_token()
```

**Value**

unsets environmental variables token (no return object)

**Examples**

```
edl_unset_token()
```

---

gdal_cloud_config	<i>Recommended GDAL configuration for cloud-based access</i>
-------------------	--

---

**Description**

Sets GDAL environmental variables to recommended optimum settings for cloud-based access.

**Usage**

```
gdal_cloud_config()
```

**Details**

Based on <https://gdalcubes.github.io/source/concepts/config.html#recommended-settings-for-cloud-access>

**Value**

sets recommended environmental variables and returns invisible TRUE if successful.

**See Also**

[gdal\\_cloud\\_unconfig\(\)](#)

**Examples**

```
gdal_cloud_config()

# remove settings:
gdal_cloud_unconfig()
```

---

gdal\_cloud\_unconfig     *Restores GDAL default configuration*

---

**Description**

Unsets GDAL environmental variables set by [gdal\\_cloud\\_config\(\)](#)

**Usage**

```
gdal_cloud_unconfig()
```

**Value**

invisible TRUE if successful.

**See Also**

[gdal\\_cloud\\_config\(\)](#)

**Examples**

```
gdal_cloud_config()

# remove settings:
gdal_cloud_unconfig()
```

---

get\_nasa\_stac\_url      *Get NASA STAC URL for a Provider*

---

**Description**

Retrieves the STAC catalog URL for a specific NASA provider

**Usage**

```
get_nasa_stac_url(provider, cloud_only = TRUE)
```

**Arguments**

provider      Character; the name of the NASA STAC provider  
cloud\_only    Logical; if TRUE (default), returns only cloud-hosted STAC catalogs

**Value**

A character string containing the STAC catalog URL for the specified provider

**See Also**

[list\\_nasa\\_stacs\(\)](#)

**Examples**

```
## Not run:  
get_nasa_stac_url("LPDAAC", cloud_only = FALSE)  
get_nasa_stac_url("LPCLLOUD")  
  
## End(Not run)
```

---

list\_nasa\_stacs      *List NASA STAC Catalogs*

---

**Description**

Retrieves available STAC catalogs from NASA's Common Metadata Repository (CMR)

**Usage**

```
list_nasa_stacs(cloud_only = TRUE)
```

**Arguments**

cloud\_only    Logical; if TRUE (default), returns only cloud-hosted STAC catalogs

**Value**

A data frame of STAC catalog urls for all STAC providers in the CMR

**See Also**

[get\\_nasa\\_stac\\_url\(\)](#)

**Examples**

```
## Not run:  
list_nasa_stacs()  
list_nasa_stacs(cloud_only = FALSE)  
  
## End(Not run)
```

---

`lpdacc_example_url`      *URL for an example of an LP DAAC COG file*

---

**Description**

URL for an example of an LP DAAC COG file

**Usage**

```
lpdacc_example_url()
```

**Value**

The URL to a Cloud-Optimized Geotiff file from the LP DAAC.

**Examples**

```
lpdacc_example_url()
```

---

<code>with_gdalcubes</code>	<i>with_gdalcubes</i>
-----------------------------	-----------------------

---

**Description**

expose any GDAL\_\* or VSI\_\* environmental variables to gdalcubes, which calls GDAL in an isolated environment and does not respect the global environmental variables.

**Usage**

```
with_gdalcubes(env = Sys.getenv())
```

**Arguments**

`env` a named vector of set environmental variables. Default is usually best, which will configure all relevant global environmental variables for gdalcubes.

**Value**

NULL, invisibly.

**Examples**

```
with_gdalcubes()
```

# Index

`collections_fetch`, 3

`edl_as_s3`, 3  
`edl_download`, 4  
`edl_extract_urls`, 5  
`edl_extract_urls()`, 10  
`edl_netrc`, 6  
`edl_netrc()`, 8, 11, 12  
`edl_revoke_token`, 7  
`edl_s3_token`, 8  
`edl_search`, 9  
`edl_set_token`, 11  
`edl_stac_urls`, 12  
`edl_unset_netrc`, 12  
`edl_unset_s3`, 13  
`edl_unset_token`, 14  
`edl_unset_token()`, 11

`gdal_cloud_config`, 14  
`gdal_cloud_config()`, 6, 13, 15  
`gdal_cloud_unconfig`, 15  
`gdal_cloud_unconfig()`, 15  
`get_nasa_stac_url`, 16  
`get_nasa_stac_url()`, 17

`httr::GET()`, 3

`list_nasa_stacs`, 16  
`list_nasa_stacs()`, 16  
`lpdacc_example_url`, 17

`rstac::collections()`, 3  
`rstac::get_request()`, 3

`Sys.unsetenv`, 14

`tools::R_user_dir()`, 5, 6, 10, 13

`with_gdalcubes`, 18