Before you start

Contents

Introduction

The `CDSE` package for R was developed to allow access to the 'Copernicus Data Space Ecosystem' data and services from R. The `'Copernicus Data Space Ecosystem'`, deployed in 2023, offers access to the EO data collection from the Copernicus missions, with discovery and download capabilities and numerous data processing tools. In particular, the 'Sentinel Hub' API provides access to the multi-spectral and multi-temporal big data satellite imagery service, capable of fully automated, real-time processing and distribution of remote sensing data and related EO products. Users can use APIs to retrieve satellite data over their AOI and specific time range from full archives in a matter of seconds. When working on the application of EO where the area of interest is relatively small compared to the image tiles distributed by Copernicus (100 x 100 km), it allows to retrieve just the portion of the image of interest rather than downloading the huge tile image file and processing it locally. The goal of the `CDSE` package is to provide easy access to this functionality from R.

The main functions allow to search the catalog of available imagery from the Sentinel-1, Sentinel-2, Sentinel-3, and Sentinel-5 missions, and to process and download the images of an area of interest and a time range in various formats. Other functions might be added in subsequent releases of the package.

Accessing CDSE data and services

Access to the `'Copernicus Data Space Ecosystem'` is free, but you have to register to use the API. You can create a free account as explained in User registration and authentication. The free account has some limitations and quotas applied to it, but it should be sufficient for most individual users. The details are provided in Quotas and Limitations.

API authentication

Most of the API functions require a specific authentication. The API uses OAuth2 Authentication and requires that you have an access token. In essence, this is a piece of information you add to your requests so the server knows it's you. To be able to request a token, you need to register an OAuth Client in your account settings. Here you will obtain your client credentials - client id and client secret. You will use these

client credentials to authenticate with the API. **Make sure to copy your personal OAuth secret, as you will not be able to see it again!** You can find more details on the documentation page dedicated to API authentication.

Storing client credentials

You should store your client credentials securely. Do not hard-code them (include as clear text) in scripts, particularly in the scripts shared with others. Don't save them to a repository (like Git) or to a shared folder. You can of course provide the credentials every time they are needed, but this is a very cumbersome approach. A simple way to keep them persistently available is to store them as system environment variables. This can be achieved by defining them in your personal or project-level `.Renviron` file. We recommend this method for its simplicity and use it in our examples. You could also set the environment variables with `Sys.setenv()`, but you should note that is not persistent; the values are lost when the R session terminates. Another option is to store them in the global `options()`, typically in your personal `.Rprofile`. These two options require the name-value pairs, for example, `CDSE_ID = "yourid"` and `CDSE_SECRET = "yoursecret"`. You can find more information about dealing with sensitive information in R at Managing secrets.

OAuth authentication client

The recommended way to authenticate with the CDSE API is to use the `httr2_oauth_client` object (from the `httr2` package) returned by the `GetOAuthClient` function, as shown below. You have to provide your client credentials as arguments to the function. The returned object should be passed as the `client` argument to the functions requiring the authentication. The underlying services in the `httr2` package will automatically take care of the authentication lifecycle management (refreshing token, etc.).

The credentials have been obfuscated in the output.

```
id <- Sys.getenv("CDSE_ID")
secret <- Sys.getenv("CDSE_SECRET")
OAuthClient <- GetOAuthClient(id = id, secret = secret)
class(OAuthClient)
#> [1] "httr2_oauth_client"
OAuthClient
#> <httr2_oauth_client>
#> name: x9x99xx99x9xx99xx99xx9xx99x99x99
#> id: xx-9x999x9x-9999-999x-xxxx-x9999x99x99x
#> secret: <REDACTED>
#> token_url: https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol/openid-connect/token
#> auth: oauth_client_req_auth_header
```

However, it should be noted that the object returned by the `GetOAuthClient` function has not been validated against the backend. The credentials provided will only be checked the first time this object is used in a query. Therefore, even if the `GetOAuthClient` function does not raise an error you cannot assume that your credentials have been accepted by the backend. To demonstrate it, you can provide a dummy id, secret, and URL, and no error will be raised at this stage.

```
id <- "my_dummy_id"
secret <- "my_dummy_secret"
OAuthClient <- GetOAuthClient(id = id, secret = secret, url = "https://my_dummy_url.org")
class(OAuthClient)
#> [1] "httr2_oauth_client"
OAuthClient
#> <httr2_oauth_client>
#> * name     : "a43e08ed613f62d6ca1af36d4230ccf6"
#> * id       : "my_dummy_id"
```

```
#> * secret   : <REDACTED>
#> * token_url: "https://my_dummy_url.org"
#> * auth     : "oauth_client_req_auth_header"
```

OAuth authentication token

In order to be able to check immediately if your credentials work correctly, we have provided another authentication function called `GetOAuthToken`. It takes the same arguments as the above mentioned `GetOAuthClient` function, but it verifies the credentials immediately. If successful, it returns a connection token, (very long) string that can be be passed as the `token` argument to the functions requiring the authentication. If your credentials have been refused by the backend, an error is raised. Please not that in this case you must explicitly take care yourself of the token lifecycle management. We therefore recommend that you use this function only to test that your credentials work, but to prefer passing the object returned by the `GetOAuthClient` as the `client` argument to the functions from the `CDSE` package that require authentication.

The token has been obfuscated and shortened in the output.

```
id <- Sys.getenv("CDSE_ID")
secret <- Sys.getenv("CDSE_SECRET")
OAuthToken <- GetOAuthToken(id = id, secret = secret)
class(OAuthToken)
#> [1] "character"
OAuthToken
#> [1] "xxXxxXxxXxXXXxX9XxXxXxX9xXXxXxXxxXXxxxx9xxXxX9XXXXx9X9xXxX9XxXXXxxX9xXXXxx......"
```

Note for Windows users

On some Windows systems, depending on the network and security settings, the things might not work out of the box. If you get an error while connecting to the CDSE API complaining about SSL/TLS handshake problem, try setting the environment variable `CURL_SSL_BACKEND` to `openssl` **before** using the functions from the `CDSE` package. You can restart R session, and type `Sys.setenv(CURL_SSL_BACKEND = "openssl")` before using the `CDSE` package. Even better, you can permanently set this environment variable in your `.Renviron` file (by adding the line `CURL_SSL_BACKEND = "openssl"`) or setting it in your Windows system settings environment variables.

You can find more information about this issue here.