Package: CopernicusDataspace (via r-universe)

May 21, 2026

Title Search Download and Handle Data from the Copernicus Data Space Ecosystem

Version 0.0.1

Description The Copernicus Data Space Ecosystem, is an open ecosystem that provides free instant access to a wide range of data and services from the Copernicus Sentinel missions and more on our planet's land, oceans and atmosphere. This package provides entry points to several APIs allowing users to access the data directly in R.

Depends R (>= 4.1.0)

Imports aws.s3, cli, dplyr, httr2, jsonlite, lubridate, memoise, methods, rlang, sf, stringr, tibble, tidyr, xml2

Suggests clipr, jose, knitr, lwgeom, rmarkdown, rstudioapi, stars, testthat (>= 3.0.0)

License GPL (>= 3)

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

Collate 'CopernicusDataspace-package.R' 'helpers.R' 'account.R' 'init.R' 'geometry.R' 'tidyverse.R' 'login.R' 's3.R' 'req_perform.R' 'odata_products.R' 'odata_bursts.R' 'sentinelhub.R' 'stac.R'

Config/testthat/edition 3

URL https://github.com/pepijn-devries/CopernicusDataspace, https://pepijn-devries.github.io/CopernicusDataspace/

BugReports https://github.com/pepijn-devries/CopernicusDataspace/issues

VignetteBuilder knitr

NeedsCompilation no

Author Pepijn de Vries [aut, cre] (ORCID: <https://orcid.org/0000-0002-7961-6646>), Alicia Hamer [rtm] (ORCID: <https://orcid.org/0000-0001-6809-622X>), LVVN (Ministerie van Landbouw, Visserij, Voedselzekerheid en Natuur) [fnd] (ROR: <https://ror.org/03b1hdw57>), WMR (Wageningen Marine Research) [ctr] (ROR: <https://ror.org/013dzwk66>)

Maintainer Pepijn de Vries <pepijn.devries@outlook.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-05-21 09:10:15 UTC

RemoteUrl https://github.com/cran/CopernicusDataspace

RemoteRef HEAD

RemoteSha 99cdb26dbda8f342e56ccec36554f6393411ed9f

Contents

`dse_access_token` *Client Information and Access Token for the Data Space Store API*

Description

To regulate server traffic, the Data Space Ecosystem uses user accounts to regulate and ensure fair usage. These functions will get you a token and information about your usage.

Usage

```
dse_access_token(
  client_id = dse_get_client_id(),
  client_secret = dse_get_client_secret(),
  ...
)

dse_public_access_token(
  username = dse_get_username(),
  password = dse_get_password()
)

dse_get_client_id(...)

dse_set_client_id(value, ...)

dse_set_username(value, ...)

dse_set_password(value, ...)

dse_get_client_secret(...)

dse_get_username(...)

dse_get_password(...)

dse_set_client_secret(value, ...)

dse_has_client_info(...)

dse_has_account(...)
```

Arguments

client_id	ID of the client registered under your account.
client_secret	Secret provided for the client registered under your account.
...	Ignored
username	Your Copernicus Data Space username (usually your e-mail address).
password	Your Copernicus Data Space password for your account.
value	Assignment value for setting username, password, client_id or client_secret as environment variable. Once set, it will persist for the remainder of the R session.

Details

Before you can use most of the Data Space Ecosystem services, you need to create an account and register as a client. This will let you retrieve an access token, that can be used for authentication purposes in API requests. It is also used to manage your usage and rate limiting (see also dse_usage() and dse_user_statistics()).

Note that Amazon Simple Storage Service (s3) has separate authentication requirements. See dse_s3_get_key() for details.

Creating an Account:

First step is creating an account. You can create one by visiting the login page and click "register". Fill out the form and follow the instructions.

Registering Client:

In order to register a client, visit the Sentinel Dashboard and go to "User settings". There you will have the option to create an OAuth client. Follow the instructions, and make sure to safely copy the client id and secret. The latter is only displayed once. More detailed instructions are provided by the API documentation

Client Details as Environment Variables:

When you share R code, you probably don't want to share your account details. You can avoid using your client_id and client_secret in your script by setting them as environment variable. You can do this yourself manually by calling dse_set_client_id() and dse_set_client_secret() at the start of each session.

You can download OData simply through https with just you username and password. You can also store those as environment variables for your convenience. If you name those CDSE_API_USERNAME and CDSE_API_PASSWORD respectively, they are picked up automatically with dse_get_username() and dse_get_password(). OData needs a public access token which is generated with dse_public_access_token(), which needs your username and password.

You can also define them in your .Rprofile file with Sys.setenv(CDSE_API_CLIENTID = "<your id>") and Sys.setenv(CDSE_API_CLIENTSECRET = "<your secret>"). This way, they are set each time you start a new R session.

The environment variables are used by default by dse_access_token(), and dse_public_access_token() so you don't have to specify the client details as arguments.

Obtain Token and Validity:

After completing the previous two steps, your are now set to obtain an access token with `dse_access_token()`.

Repeatedly requesting an access token may invoke rate limiting measures. Therefore, this package uses caching to temporarily store the access token during each R session. Calling `dse_access_token()` will therefore only contact the server once for a token for each unique combination of `client_id` and `client_server`. After that, the cached result will be reused during the session

There is a catch: the token provided by the server is only valid for a limited time (usually 30 minutes). So, when the token has expired, you need to wipe the cache. You can do so by calling `memoise::forget(dse_access_token)` or restarting the R session.

Value

In case of `dse_get_client_id()` and `dse_get_client_secret()`, you can get (or set) client details as environment variables. This way, they will persist throughout your R session.

The function `dse_has_client_info()` returns a `logical` value, indicating whether client details (id and secret) are available as environmental variable. Note that if this function returns `TRUE`, it doesn't guarantee that the details are valid (just that they are available).

In case of `dse_access_token()` and `dse_public_access_token()` a named `list` is returned, containing the access token (named `"token"`) and some additional meta information.

References

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Overview/Authentication.html

See Also

`dse_usage()`

`dse_user_statistics()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_set_gdal_token()`, `dse_usage()`

Other authentication: dse_get_token_details(), dse_has_s3_secret(), dse_set_gdal_token(), dse_usage()

Examples

```
if (interactive() && dse_has_client_info()) {
  token <- dse_access_token()
}
if (interactive() && dse_has_account()) {
  token_public <- dse_public_access_token()
}
```

---

dse_get_token_details *Decode Access Token*

Description

This function decodes an access token and returns a named list with information from the token.

Usage

```
dse_get_token_details(token = dse_access_token())
```

Arguments

token	A token obtained with dse_access_token() or dse_public_access_token().

Value

A named list with token info

See Also

Other authentication: dse_access_token(), dse_has_s3_secret(), dse_set_gdal_token(), dse_usage()

Examples

```
if (interactive() && dse_has_client_info()) {
  dse_get_token_details()
}
```

`dse_has_s3_secret` *Setup Amazon Simple Storage Service for the Data Space Ecosystem*

Description

using Amazon Simple Storage Service (s3) in the Data Space Ecosystem requires a key and secret. These functions help you managing these details and setting up an s3 client.

Usage

```
dse_has_s3_secret()

dse_s3_get_key(...)

dse_s3_set_key(value, ...)

dse_s3_get_secret(...)

dse_s3_set_secret(value, ...)
```

Arguments

`...`	Ignored
`value`	Replacement value for the `s3_key` or `s3_secret`.

Details

Working with s3 in the Data Space Ecosystem requires you to create an account, then register an s3 key as described below. Note that the SentinelHub requires a different authentication method. See `dse_access_token()` for more details on that.

Creating an Account:

First step is creating an account. You can create one by visiting the login page and click "register". Fill out the form and follow the instructions.

Registering a s3 Key:

Now that you have an account, you should visit the s3-credentials page, and log in with your account details. By clicking "add credential", you can create a new key and secret. Store them in a safe place, as the secret is only shown once. You can pass the key and secret as `s3_key` and `s3_secret` arguments to functions requesting them. You can also store them as environment variables such that they persist throughout the R session and don't have to be passed as arguments (see below).

S3 Key and Secret as Environment Variables:

When you share R code, you probably don't want to share your account details. You can avoid using your `s3_key` and `s3_secret` in your script by setting them as environment variable. You

can do this yourself manually by calling dse_s3_set_key() and dse_s3_set_secret() at the start of each session.

You can also define them in your .Rprofile file with Sys.setenv(CDSE_API_S3ID = "<your key>") and Sys.setenv(CDSE_API_S3SECRET = "<your secret>"). This way, they are set each time you start a new R session.

Value

dse_s3_get_key() and dse_s3_get_secret() will return the requested s3 details if set as environment variable (see details).

dse_has_s3_secret() returns a logical value indicating whether s3 details (key and secret) are set. It will not determine whether the details are valid.

References

https://documentation.dataspace.copernicus.eu/APIs/S3.html

See Also

Other authentication: dse_access_token(), dse_get_token_details(), dse_set_gdal_token(), dse_usage()

Other s3: dse_s3_download(), dse_s3_set_gdal_options(), dse_s3_uri_to_vsi()

Other authentication: dse_access_token(), dse_get_token_details(), dse_set_gdal_token(), dse_usage()

Other s3: dse_s3_download(), dse_s3_set_gdal_options(), dse_s3_uri_to_vsi()

Other authentication: dse_access_token(), dse_get_token_details(), dse_set_gdal_token(), dse_usage()

Other s3: dse_s3_download(), dse_s3_set_gdal_options(), dse_s3_uri_to_vsi()

Other authentication: dse_access_token(), dse_get_token_details(), dse_set_gdal_token(), dse_usage()

Other s3: dse_s3_download(), dse_s3_set_gdal_options(), dse_s3_uri_to_vsi()

Other authentication: dse_access_token(), dse_get_token_details(), dse_set_gdal_token(), dse_usage()

Other s3: dse_s3_download(), dse_s3_set_gdal_options(), dse_s3_uri_to_vsi()

---

dse_odata_attributes *List OData Attributes*

Description

Collect a list of OData attributes that can be used for filtering products with dse_odata_products_request().

Usage

```
dse_odata_attributes(...)
```

Arguments

...	Ignored

Value

A `data.frame` listing all attributes for each collection.

See Also

Other odata: `dse_odata_bursts_request()`, `dse_odata_download()`, `dse_odata_download_path()`, `dse_odata_product_nodes()`, `dse_odata_products_request()`, `dse_odata_quicklook()`

Examples

```
if (interactive()) {
  dse_odata_attributes()
}
```

---

dse_odata_bursts_request

Create a OData Request for a Data Space Ecosystem Bursts Data

---

Description

Obtain metadata for burst data associated with specific products.

Usage

```
dse_odata_bursts_request(...)

dse_odata_bursts(...)
```

Arguments

...	Ignored

Details

For more details about bursts check the burst API documentation.

You can apply some tidyverse functions (see tidy_verbs) to `odata_request` object returned by `dse_odata_bursts_request()`. These apply lazy evaluation. Meaning that they are just added to the object and are only evaluated after calling either `dplyr::compute()` or `dplyr::collect()` (see examples).

Value

Returns a `data.frame` with burst information.

See Also

Other odata: dse_odata_attributes(), dse_odata_download(), dse_odata_download_path(), dse_odata_product_nodes(), dse_odata_products_request(), dse_odata_quicklook()

Other odata: dse_odata_attributes(), dse_odata_download(), dse_odata_download_path(), dse_odata_product_nodes(), dse_odata_products_request(), dse_odata_quicklook()

Examples

```
if (interactive() && dse_has_s3_secret()) {
  dse_odata_bursts(ParentProductId == "879d445c-2c67-5b30-8589-b1f478904269")

  burst_req <-
    dse_odata_bursts_request(ParentProductId == "879d445c-2c67-5b30-8589-b1f478904269")

  ## Note that these are large files and may take a while to download:
  dse_odata_download(
    burst_req,
    tempdir()
  )
}
```

---

dse_odata_download *Download Data Space Ecosystem Products Through OData API*

Description

Use dse_odata_products() or dse_odata_products_request(), dse_odata_bursts() or dse_odata_bursts_request to find a product or burst information. Use this function to download the product(s) or burst information.

Usage

```
dse_odata_download(
  request,
  destination,
  ...,
  s3_key = dse_s3_get_key(),
  s3_secret = dse_s3_get_secret()
)
```

Arguments

request	A request containing products or burst data that you wish to download. Use dse_odata_products_request() or dse_odata_bursts_request() to formulate product or burst specifications.
destination	A character string specifying the directory path, where to store downloaded products

`...` Ignored.

`s3_key, s3_secret`
: The s3 key and secret registered under your Data Space Ecosystem account

Value

A vector of downloaded file names stored at `destination`

See Also

Other odata: `dse_odata_attributes()`, `dse_odata_bursts_request()`, `dse_odata_download_path()`, `dse_odata_product_nodes()`, `dse_odata_products_request()`, `dse_odata_quicklook()`

Examples

```
if (interactive() && dse_has_s3_secret()) {
  dse_odata_download(
    dse_odata_products(Name == "S1C_AUX_PP2_V20241204T000000_G20251024T110034.SAFE"),
    destination = tempdir())
  dse_odata_download(
    dse_odata_products(
      Name ==
        "S1A_IW_OCN__2SDH_20250707T210608_20250707T210625_059983_07739B_893E.SAFE"),
    destination = tempdir())
  dse_odata_download(
    dse_odata_products(
      Id %in% c("c8ed8edb-9bef-4717-abfd-1400a57171a4",
                "86288a07-560c-364f-b8ce-669d95f06fa0")),
    destination = tempdir())
}
```

---

dse_odata_download_path

Alternative Route to Download OData Products

Description

Downloading data using the OData API is probably fastest by using `dse_odata_download()`. As an alternative, you can use this function which uses the https protocol to download a product.

Usage

```
dse_odata_download_path(
  product,
  node_path = "",
  destination,
  progress = TRUE,
  ...,
  token = dse_public_access_token()
)
```

Arguments

product	Hexadecimal id of the product to be downloaded
node_path	Path to a specific file in the product. When left blank ("") The function will attempt to download the entire product as a zip archive.
destination	Path to a directory where to store the downloaded file
progress	logical value. If TRUE shows download progress.
...	Ignored
token	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with dse_access_token() (default). Without a valid token you will likely get an "access denied" error.

Value

Returns a httr2_response class object. It's body will hold the filename of the downloaded file

See Also

Other odata: dse_odata_attributes(), dse_odata_bursts_request(), dse_odata_download(), dse_odata_product_nodes(), dse_odata_products_request(), dse_odata_quicklook()

Examples

```
if (interactive() && dse_has_account()) {

  dse_odata_download_path(
    product     = "2f497806-0101-5eea-83fa-c8f68bc56b0c",
    node_path   =
      paste("DEM1_SAR_DTE_90_20101213T034716_20130408T035028_ADS_000000_5033.DEM",
            "Copernicus_DSM_30_S09_00_E026_00", "DEM",
            "Copernicus_DSM_30_S09_00_E026_00_DEM.dt1", sep = "/"),
    destination = tempdir()
  )

  dse_odata_download_path(
    product     = "ce4576eb-975b-40ff-8319-e04b00d8d444",
    destination = tempdir())

}
```

---

dse_odata_product_nodes

List OData Product Nodes (i.e. Files and Directories)

Description

If you know the product Id, you can use this function to retrieve information about nodes (i.e. files and directories) within the product.

Usage

```
dse_odata_product_nodes(product, node_path = "", recursive = FALSE, ...)
```

Arguments

product	A product identifier (Id)
node_path	Path of nodes separated by forward slashes ("/"). Path for which to list nodes. Default is "", which is the root of the product
recursive	A logical value. If set to TRUE, it will recursively list all nested nodes. Default is FALSE.
...	Ignored

Value

A data.frame with information on the requested node(s)

See Also

Other odata: dse_odata_attributes(), dse_odata_bursts_request(), dse_odata_download(), dse_odata_download_path(), dse_odata_products_request(), dse_odata_quicklook()

Examples

```
if (interactive()) {
  nodes <- dse_odata_product_nodes("c8ed8edb-9bef-4717-abfd-1400a57171a4")
  nodes <- dse_odata_product_nodes("c8ed8edb-9bef-4717-abfd-1400a57171a4",
                                   recursive = TRUE)
}
```

---

dse_odata_products_request

Create a OData Request for a Data Space Ecosystem Product

Description

OData is an application programming interface (API) used to disseminate Copernicus Data Space Ecosystem products. This function creates a request for this service, which can be used to obtain a data.frame with product information. This request supports several tidyverse methods for filtering and arranging the data.

Usage

```
dse_odata_products_request(..., expand)

dse_odata_products(..., expand = NULL)
```

Arguments

...	Ignored in case of dse_odata_products_request(). Dots are passed to embedded dplyr::filter() in case of dse_odata_products()
expand	Additional information to be appended to the result. Should be any of "Attributes", "Assets", or "Locations". Note that, these columns are not affected by dplyr::select() calls (before calling dplyr::collect()).

Details

You can apply some tidyverse functions (see tidy_verbs) to odata_request object returned by dse_odata_products_request(). These apply lazy evaluation. Meaning that they are just added to the object and are only evaluated after calling either dplyr::compute() or dplyr::collect() (see examples).

Value

Returns an odata_request class object in case of dse_odata_products_request(), which is an extension of httr2::request(). In case of dse_odata_products() a data.frame listing requested products is returned.

References

https://documentation.dataspace.copernicus.eu/APIs/OData.html

See Also

Other odata: dse_odata_attributes(), dse_odata_bursts_request(), dse_odata_download(), dse_odata_download_path(), dse_odata_product_nodes(), dse_odata_quicklook()

Other odata: dse_odata_attributes(), dse_odata_bursts_request(), dse_odata_download(), dse_odata_download_path(), dse_odata_product_nodes(), dse_odata_quicklook()

Examples

```
if (interactive()) {
  bbox <-
    sf::st_bbox(
      c(xmin = 5.261, ymin = 52.680, xmax = 5.319, ymax = 52.715),
      crs = 4326) |>
      sf::st_as_sfc()

  dse_odata_products_request() |>
    dplyr::filter(
      `ContentDate/Start` > "2025-01-01") |>
    sf::st_intersects(bbox) |>
    dplyr::arrange(dplyr::desc(Id)) |>
    dplyr::slice_head(n = 100) |>
    dplyr::collect()
}
```

dse_odata_quicklook *Download a Quicklook for a Product*

Description

Downloads a 'quicklook' for a product. If the `rstudioapi` package is installed, it will attempt to open the image in the Viewer panel.

Usage

```
dse_odata_quicklook(product, destination, ...)
```

Arguments

`product`	Identifier (Id) for the product for which to obtain a quicklook.
`destination`	A destination path where to store the image.
`...`	Ignored

Value

Returns `NULL` invisibly

See Also

Other odata: `dse_odata_attributes()`, `dse_odata_bursts_request()`, `dse_odata_download()`, `dse_odata_download_path()`, `dse_odata_product_nodes()`, `dse_odata_products_request()`

Examples

```
if (interactive()) {
  dse_odata_quicklook(
    "91822f33-b15c-5b60-aa39-6d9f6f5c773b",
    tempfile(fileext = ".jpg"))
}
```

dse_s3_download *Download Asset Through Uniform Resource Identifier*

Description

When the Uniform Resource Identifier (URI, starting with "s3://") for an asset is known, this function can be used to download it

Usage

```
dse_s3_download(
  uri,
  destination,
  ...,
  s3_key = dse_s3_get_key(),
  s3_secret = dse_s3_get_secret()
)
```

Arguments

uri	A Uniform Resource Identifier (URI, starting with "s3://"). You can look for them in the STAC catalogue, either using a web browser or dse_stac_search_request() (see example).
destination	Destination path to a directory where to store the downloaded file(s)
...	Ignored
s3_key, s3_secret	The s3 key and secret registered under your Data Space Ecosystem account

Value

A vector of file names stored at destination

See Also

Other s3: dse_has_s3_secret(), dse_s3_set_gdal_options(), dse_s3_uri_to_vsi()

Examples

```
if (interactive() && dse_has_s3_secret()) {
  library(dplyr)

  ## Retrieve a URI for a specific asset through the STAC
  ## catalogue:
  my_uri <-
    dse_stac_search_request(
      ids = "S2A_MSIL1C_20260109T132741_N0511_R024_T39XVL_20260109T142148") |>
        select("assets.B01.href") |>
        arrange("id") |>
        collect() |>
        pull("assets") |>
        unlist()

  dse_s3_download(my_uri, tempdir())

}
```

dse_s3_set_gdal_options *Set-up S3 Configuration for GDAL Library*

Description

This function sets system environment variables, such that the GDAL library can access the Copernicus Data Space Ecosystem S3 storage. Note that these settings can be used by any package depending on the GDAL library. Most notably: `stars`, `terra`, and `gdalraster`.

Usage

```
dse_s3_set_gdal_options(
  region = "us-east-1",
  ...,
  s3_key = dse_s3_get_key(),
  s3_secret = dse_s3_get_secret()
)
```

Arguments

`region`	AWS Region used in instantiating the S3 client
`...`	Ignored
`s3_key, s3_secret`	The s3 key and secret registered under your Data Space Ecosystem account

Value

Returns a `logical` value. `TRUE` if all variables were successfully set. `FALSE` otherwise.

See Also

Other s3: `dse_has_s3_secret()`, `dse_s3_download()`, `dse_s3_uri_to_vsi()`

Examples

```
if (interactive() && dse_has_s3_secret() &&
    requireNamespace("stars")) {
  library(dplyr)

  ## Get a Virtual System Interface to a tiff file:
  vsi <-
    dse_stac_get_uri(
      "S1A_IW_GRDH_1SDV_20241125T055820_20241125T055845_056707_06F55C_12F9_COG",
      "vh", "sentinel-1-grd") |>
      dse_s3_uri_to_vsi()

  ## Make sure to set gdal options with required S3 settings
```

```
  dse_s3_set_gdal_options()

  ## You can now read the file directly from the online storage
  ## without having to download it completely:
  cog <- stars::read_stars(vsi)

  ## You can also easily plot a downsampled version
  plot(cog, downsample = 50)
}
```

`dse_s3_uri_to_vsi` *Convert Uniform Resource Identifier to Virtual System Identifier*

Description

Convert Uniform Resource Identifier (URI) to Virtual System Identifier (VSI). The Copernicus Data Space Ecosystem returns URIs for accessing assets. Packages that use the GDAL library (e.g., `stars`, `terra` and `gdalraster`) can use VSI to access raster data directly. Use this function to convert such an URI to a VIS.

Usage

```
dse_s3_uri_to_vsi(uri, streaming = TRUE)
```

Arguments

`uri`	A Uniform Resource Identifier, pointing to an S3 storage file. You can retrieve one with `dse_stac_get_uri()`.
`streaming`	A `logical` value that allows to toggle between `"\\vsis3\\"` and `"\\vsis3_streaming\\"` (default). The latter is faster for reading files from its resource, but does not allow random access. The first supports random access, but is not as fast at reading.

Value

A `character` string representing the VSI

See Also

Other s3: `dse_has_s3_secret()`, `dse_s3_download()`, `dse_s3_set_gdal_options()`

Examples

```
if (interactive()) {
  dse_stac_get_uri(
    "S1A_IW_GRDH_1SDV_20241125T055820_20241125T055845_056707_06F55C_12F9_COG",
    "vh", "sentinel-1-grd") |>
    dse_s3_uri_to_vsi()
}
```

dse_set_gdal_token *Set Copernicus Data Space Ecosystem Access Token for GDAL Driver*

Description

This function sets system environment variables, such that the GDAL library can access the Copernicus Data Space Ecosystem https storage. Note that these settings can be used by any package depending on the GDAL library. Most notably: `stars`, `terra`, and `gdalraster`.

Usage

```
dse_set_gdal_token(token = dse_access_token())
```

Arguments

`token`	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with `dse_access_token()` (default). Without a valid token you will likely get an "access denied" error.

Value

Returns a `logical` value. `TRUE` if all variables were successfully set. `FALSE` otherwise.

See Also

Other authentication: `dse_access_token()`, `dse_get_token_details()`, `dse_has_s3_secret()`, `dse_usage()`

Examples

```
if (interactive() && dse_has_client_info() &&
    requireNamespace("stars")) {
  uri <-
    dse_stac_get_uri(
      "S2A_MSIL1C_20260109T132741_N0511_R024_T39XVL_20260109T142148",
      "B01", type = "odata")

  dse_set_gdal_token()

  ## As this URI is zipped, it need to be downloaded.
  ## But you can access it directly:
  jp2 <- stars::read_stars(uri)
}
```

dse_sh_collections *List Sentinel Hub Collections*

Description

List collections that are available from the Sentinel Hub.

Usage

```
dse_sh_collections(...)
```

Arguments

...	Ignored

Value

Returns a `data.frame` with information about the collections available from the Sentinel Hub

See Also

Other sentinelhub: `dse_sh_custom_scripts()`, `dse_sh_features()`, `dse_sh_get_custom_script()`, `dse_sh_prepare_input()`, `dse_sh_process()`, `dse_sh_queryables()`, `dse_sh_search_request()`, `dse_sh_use_requests_builder()`

Examples

```
if (interactive()) {
  dse_sh_collections()
}
```

dse_sh_custom_scripts *List Custom JavaScripts for Processing Sentinel Hub Data*

Description

Custom Eval Scripts, that can be used in Sentinel Hub requests, for processing data. This functions lists scripts available from https://github.com/sentinel-hub/custom-scripts. They can be retrieved with `dse_sh_get_custom_script()`.

Usage

```
dse_sh_custom_scripts(...)
```

Arguments

...	Ignored

Details

Make sure that you have sufficient monthly quota left to process images. You can check with dse_usage().

Value

Returns a data.frame with custom scripts, containing a column with a title and one with a relative URL.

References

- https://custom-scripts.sentinel-hub.com/
- https://github.com/sentinel-hub/custom-scripts

See Also

Other sentinelhub: dse_sh_collections(), dse_sh_features(), dse_sh_get_custom_script(), dse_sh_prepare_input(), dse_sh_process(), dse_sh_queryables(), dse_sh_search_request(), dse_sh_use_requests_builder()

Examples

```
if (interactive()) {
  dse_sh_custom_scripts()
}
```

---

dse_sh_features *List Sentinel Hub Features*

Description

List Sentinel Hub features for a specified period and region.

Usage

```
dse_sh_features(
  collection,
  bbox,
  datetime,
  limit = 10,
  ...,
  token = dse_access_token()
)
```

Arguments

collection	A collection for which to list the features. See dse_sh_collections() for a list of Sentinel Hub collections.
bbox	An object that can be converted into a bbox class object (see sf::st_bbox()).
datetime	A date-time object, or a vector of two date time objects (in case of a range). Or an object that can be converted into a datetime object.
limit	The number of records to which the output is limited. Should be between 1 and 100, and defaults to 10.
...	Ignored
token	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with dse_access_token() (default). Without a valid token you will likely get an "access denied" error.

Value

Returns a data.frame listing features available on SentinelHub for processing.

See Also

Other sentinelhub: dse_sh_collections(), dse_sh_custom_scripts(), dse_sh_get_custom_script(), dse_sh_prepare_input(), dse_sh_process(), dse_sh_queryables(), dse_sh_search_request(), dse_sh_use_requests_builder()

Examples

```
if (interactive() && dse_has_client_info()) {
  dse_sh_features(
    collection = "sentinel-2-l2a",
    bbox       = c(5.261, 52.680, 5.319, 52.715),
    datetime   = c("2025-01-01 UTC", "2025-01-07 UTC"))
}
```

---

dse_sh_get_custom_script

Retrieve Custom JavaScripts to be Used by Sentinel Hub

Description

Sentinel Hub uses JavaScripts to process satellite images. There is a repository with such custom scripts. They can be listed with dse_sh_custom_scripts(). Use the relative URL (relUrl) from that list to obtain the actual script with this function.

Usage

```
dse_sh_get_custom_script(rel_url)
```

Arguments

rel_url	A relative URL found with dse_sh_custom_scripts().

Value

A character string containing JavaScript code. This script can be used with dse_sh_process()

See Also

Other sentinelhub: dse_sh_collections(), dse_sh_custom_scripts(), dse_sh_features(), dse_sh_prepare_input(), dse_sh_process(), dse_sh_queryables(), dse_sh_search_request(), dse_sh_use_requests_builder()

Examples

```
if (interactive()) {
  dse_sh_get_custom_script("/sentinel-2/tonemapped_natural_color/")
}
```

---

dse_sh_prepare_input *Prepare Input and Output Fields for Sentinel Hub Request*

Description

dse_sh_process() requires a named list for input and output settings. The functions documented here produce those lists required by such a process request.

Usage

```
dse_sh_prepare_input(
  bounds,
  time_range,
  collection_name = "sentinel-2-l2a",
  id = NA,
  max_cloud_coverage = 100,
  mosaicking_order = "default",
  upsampling = "default",
  downsampling = "default",
  harmonize_values = FALSE,
  ...
)

dse_sh_prepare_output(
  width = 512,
  height = 512,
  output_format = "tiff",
  bbox,
  ...
)
```

Arguments

bounds	A bounding box or geometry (classes sf::bbox, sf::sf, sf::sfc) defining the boundaries of the output image.
time_range	A vector of two date-time values, specifying the time range for satellite data to include in the process.
collection_name	A collection name. defaults to "sentinel-2-l2a" to ensure you get Sentinel-2 L2A data.
id	An identifier. Not documented by the API reference material.
max_cloud_coverage	Maximum cloud cover to be included in the process. Value between 0 and 100 (default) percent.
mosaicking_order	Sets the order of overlapping tiles from which the output result is mosaicked. Should be any of "default", "mostRecent", "leastRecent", or "leastCC". See also the API documentation.
upsampling, downsampling	Specify the interpolation technique when the output resolution is smaller or larger respectively than the available source data. See also the API documentation.
harmonize_values	A logical value indicating whether units are harmonised as indicated in the API documentation.
...	Ignored
width, height	Size of the output image in pixels. These are ignored if bbox is specified.
output_format	File format for the output file. Should be one of "tiff" (default), "jpeg", "png", or "json".
bbox	You can optionally provide a bounding box (i.e., a copy of bounds) to calculate width and height with fixed aspect ratio. Width will be 512 be definition, the height is chosen such that it matches with the bounding box

Value

A named list that can be used as input and output argument to dse_sh_process().

References

- https://shapps.dataspace.copernicus.eu/requests-builder/

See Also

dse_sh_process()

Other sentinelhub: dse_sh_collections(), dse_sh_custom_scripts(), dse_sh_features(), dse_sh_get_custom_script(), dse_sh_process(), dse_sh_queryables(), dse_sh_search_request(), dse_sh_use_requests_builder()

Other sentinelhub: dse_sh_collections(), dse_sh_custom_scripts(), dse_sh_features(), dse_sh_get_custom_script(), dse_sh_process(), dse_sh_queryables(), dse_sh_search_request(), dse_sh_use_requests_builder()

Examples

```
dse_sh_prepare_input(
  bounds = c(5.261, 52.680, 5.319, 52.715),
  time_range = c("2025-06-01 UTC", "2025-07-01 UTC")
)

library(sf)
shape <- st_bbox(c(xmin = 5.261, ymin = 52.680,
                   xmax = 5.319, ymax = 52.715), crs = 4326) |>
           st_as_sfc()
dse_sh_prepare_input(
  bounds = shape,
  time_range = c("2025-06-01 UTC", "2025-07-01 UTC")
)

dse_sh_prepare_output(bbox = shape)
```

---

dse_sh_process *Process Satellite Data and Download Result*

Description

Users can request raw satellite data, simple band combinations such as false colour composites, calculations of simple remote sensing indices like NDVI, or more advanced processing such as calculation of Leaf area index (LAI).

Usage

```
dse_sh_process(
  input,
  output,
  evalscript,
  destination,
  ...,
  token = dse_access_token()
)
```

Arguments

input	A named list specifying the input satellite data to be processed with evalscript to an image. A correctly formatted list can be created with dse_sh_prepare_input().
output	A named list specifying the how to present the output image, create with evalscript. A correctly formatted list can be created with dse_sh_prepare_output().

evalscript	A character string containing a piece of JavaScript, that will be run on the Sentinel Hub server. It is used to translate satellite data to pixel data in a georeferenced image. For more information on setting up such a script please consult the API documentation. You can also use dse_sh_get_custom_script() to obtain ready-to-go scripts from the SentinelHub repository.
destination	A file name to store the downloaded image.
...	Ignored
token	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with dse_access_token() (default). Without a valid token you will likely get an "access denied" error.

Details

Use dse_sh_use_requests_builder() if you want to use the graphical user interface at Sentinel Requests Builder. to define a request.

Value

A httr2_response class object containing the location of the downloaded file at its destination.

References

- https://docs.sentinel-hub.com/api/latest/api/process/
- https://shapps.dataspace.copernicus.eu/requests-builder/
- https://custom-scripts.sentinel-hub.com/
- https://github.com/sentinel-hub/custom-scripts
- https://docs.sentinel-hub.com/api/latest/evalscript/

See Also

Other sentinelhub: dse_sh_collections(), dse_sh_custom_scripts(), dse_sh_features(), dse_sh_get_custom_script(), dse_sh_prepare_input(), dse_sh_queryables(), dse_sh_search_request(), dse_sh_use_requests_builder()

Examples

```
if (interactive() && dse_has_client_info()) {

  bounds <- c(5.261, 52.680, 5.319, 52.715)

  ## prepare input data
  input <-
    dse_sh_prepare_input(
      bounds = bounds,
      time_range = c("2025-06-01 UTC", "2025-07-01 UTC")
    )

  ## prepare ouput format
```

```
  output <- dse_sh_prepare_output(bbox = bounds)

  ## retrieve processing script
  evalscript <- dse_sh_get_custom_script("/sentinel-2/l2a_optimized/")

  fl <- tempfile(fileext = ".tiff")
  ## send request and download result:
  dse_sh_process(input, output, evalscript, fl)

  if (requireNamespace("stars")) {
    library(stars)
    enkhuizen <- read_stars(fl) |> suppressWarnings()
    plot(enkhuizen, rgb = 1:3, axes = TRUE, main = "Enkhuizen")
  }
}
```

`dse_sh_queryables` *List Queryable Fields on Sentinel Hub*

Description

Return queryable fields for a specific collection on Sentinel Hub. This is useful information when composing a query with `dse_sh_prepare_input()`. Use `dse_sh_collections()` to list available collections.

Usage

```
dse_sh_queryables(collection, ..., token = dse_access_token())
```

Arguments

`collection`	Collection id for which to obtain queryable fields.
`...`	Ignored.
`token`	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with `dse_access_token()` (default). Without a valid token you will likely get an "access denied" error.

Value

Returns a named `list`, with information about queryable fields for the specified `collection`.

See Also

Other sentinelhub: `dse_sh_collections()`, `dse_sh_custom_scripts()`, `dse_sh_features()`, `dse_sh_get_custom_script()`, `dse_sh_prepare_input()`, `dse_sh_process()`, `dse_sh_search_request()`, `dse_sh_use_requests_builder()`

Examples

```
if (interactive() && dse_has_client_info()) {
  dse_sh_queryables("sentinel-2-l1c")
}
```

`dse_sh_search_request` *Create a Request for the SentinelHub Catalogue*

Description

In order to perform a search using the STAC API, you first need to create a request using `dse_sh_search_request()`. This creates a `httr2::request()` to which tidy verbs `?tidy_verbs` can be applied (e.g., `dplyr::select()` and `dplyr::filter()`. Results are retrieved by calling `dplyr::collect()` on the request.

Usage

```
dse_sh_search_request(
  collection,
  bbox,
  datetime,
  ...,
  token = dse_access_token()
)
```

Arguments

`collection`	A collection for which to list the features. See `dse_sh_collections()` for a list of Sentinel Hub collections.
`bbox`	An object that can be converted into a bbox class object (see `sf::st_bbox()`).
`datetime`	A date-time object, or a vector of two date time objects (in case of a range). Or an object that can be converted into a datetime object.
`...`	Ignored
`token`	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with `dse_access_token()` (default). Without a valid token you will likely get an "access denied" error.

Value

Returns a `sentinel_request` class object, which inherits from the `httr2::request` class. Call `dplyr::collect()` on it to retrieve results.

References

- https://docs.sentinel-hub.com/api/latest/api/catalog/

See Also

Other sentinelhub: dse_sh_collections(), dse_sh_custom_scripts(), dse_sh_features(), dse_sh_get_custom_script(), dse_sh_prepare_input(), dse_sh_process(), dse_sh_queryables(), dse_sh_use_requests_builder()

Examples

```
if (interactive() && dse_has_client_info()) {
  library(dplyr)

  dse_sh_search_request(
    collection = "sentinel-2-l2a",
    bbox       = c(5.261, 52.680, 5.319, 52.715),
    datetime   = c("2025-01-01 UTC", "2025-01-31 UTC")
  ) |>
    filter(`eo:cloud_cover` <= 10) |>
    collect()
}
```

---

dse_sh_use_requests_builder

Use Requests Builder to Send Processing Request to SentinelHub

---

Description

Use Sentinel Requests Builder to compose a request. Copy the text from the 'Request Preview' panel and submit with this function. Use dse_sh_process() when you want to define a request in R, without using a web browser.

Usage

```
dse_sh_use_requests_builder(
  build,
  destination,
  ...,
  token = dse_access_token()
)
```

Arguments

build	A character string copied from the Request Preview panel at Sentinel Requests Builder. See system.file("requests-builder.txt", package = "CopernicusDataspace") for an example of such a text. When you omit this argument, this function will attempt to retrieve the text from the system's clipboard.
destination	A file name to store the downloaded image.
...	Ignored
token	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with dse_access_token() (default). Without a valid token you will likely get an "access denied" error.

Value

A `httr2_response` class object obtained after sending the request.

References

- https://shapps.dataspace.copernicus.eu/requests-builder/

See Also

Other sentinelhub: `dse_sh_collections()`, `dse_sh_custom_scripts()`, `dse_sh_features()`, `dse_sh_get_custom_script()`, `dse_sh_prepare_input()`, `dse_sh_process()`, `dse_sh_queryables()`, `dse_sh_search_request()`

Examples

```
## Read text copied from 'Request Preview' panel on
## <https://shapps.dataspace.copernicus.eu/requests-builder/:
requests_builder <-
  system.file("requests-builder.txt", package = "CopernicusDataspace") |>
    readLines(warn = FALSE) |>
    paste(collapse = "\n")

if (interactive() && dse_has_client_info()) {
  dest <- tempfile(fileext = ".tiff")
  dse_sh_use_requests_builder(requests_builder, destination = dest)
}
```

---

dse_stac_client *Obtain Information About the STAC Client*

Description

Returns information about the STAC client used in the Data Space Ecosystem

Usage

```
dse_stac_client(...)
```

Arguments

`...`	Ignored

Value

Returns a `data.frame` with the requested information

See Also

Other stac: dse_stac_collections(), dse_stac_download(), dse_stac_get_uri(), dse_stac_guess_collection(), dse_stac_queryables(), dse_stac_search_request()

Examples

```
if (interactive()) {
  dse_stac_client()
}
```

---

dse_stac_collections *Get a Summary of all Data Space Ecosystem Collections*

Description

Use the STAC API to get a summary of all collections available from the interface.

Usage

```
dse_stac_collections(collection, ...)
```

Arguments

collection	A specific collection for which to obtain summary information. If missing (default), all collections are returned.
...	Ignored

Value

Returns a data.frame with the requested information

See Also

Other stac: dse_stac_client(), dse_stac_download(), dse_stac_get_uri(), dse_stac_guess_collection(), dse_stac_queryables(), dse_stac_search_request()

Examples

```
if (interactive()) {
  dse_stac_collections()
  dse_stac_collections("sentinel-2-l2a")
}
```

dse_stac_download *Download Asset From STAC Catalogue*

Description

Use dse_stac_search_request() to identify assets that can be downloaded. Use dse_stac_download() to download an asset by its STAC id and asset name.

Usage

```
dse_stac_download(
  asset_id,
  asset,
  collection = dse_stac_guess_collection,
  destination,
  ...,
  s3_key = dse_s3_get_key(),
  s3_secret = dse_s3_get_secret(),
  token = dse_public_access_token()
)
```

Arguments

asset_id	STAC id, used for locating the asset download details.
asset	Name of the asset to download
collection	The identifier for a collection. The default argument is the dse_stac_guess_collection() function which tries to guess the collection id from the asset_id. A more rigid approach is to provide the collection id as a character string.
destination	Directory path where to store the downloaded file.
...	Ignored
s3_key, s3_secret	The s3 key and secret registered under your Data Space Ecosystem account
token	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with dse_access_token() (default). Without a valid token you will likely get an "access denied" error.

Value

Returns the path to the downloaded file.

See Also

Other stac: dse_stac_client(), dse_stac_collections(), dse_stac_get_uri(), dse_stac_guess_collection(), dse_stac_queryables(), dse_stac_search_request()

Examples

```
if (interactive() && (dse_has_s3_secret() || dse_has_client_info())) {
  dse_stac_download(
    asset_id = "S2A_MSIL1C_20260109T132741_N0511_R024_T39XVL_20260109T142148",
    asset = "B01",
    destination = tempdir()
  )
}
```

`dse_stac_get_uri` *Get a Uniform Resource Identifier (URI) for an Asset in a Product*

Description

Get a Uniform Resource Identifier (URI) for an asset in a product. This can be used to download a file manually or connect to the asset directly straight from the source.

Usage

```
dse_stac_get_uri(
  asset_id,
  asset,
  collection = dse_stac_guess_collection,
  type = "s3",
  ...
)
```

Arguments

`asset_id`	STAC id, used for locating the asset download details.
`asset`	Name of the asset to download
`collection`	The identifier for a collection. The default argument is the `dse_stac_guess_collection()` function which tries to guess the collection id from the `asset_id`. A more rigid approach is to provide the collection id as a `character` string.
`type`	Which type of URI should be returned? Defaults to `"s3"`. Use `"odata"` to get the alternative https URI.
`...`	Ignored

Value

Returns the URI as a `character` string. If available, the local path for an asset is returned as attribute.

See Also

Other stac: `dse_stac_client()`, `dse_stac_collections()`, `dse_stac_download()`, `dse_stac_guess_collection()`, `dse_stac_queryables()`, `dse_stac_search_request()`

Examples

```
if (interactive()) {
  dse_stac_get_uri(
    asset_id = "S2A_MSIL1C_20260109T132741_N0511_R024_T39XVL_20260109T142148",
    asset = "B01"
  )
}
```

`dse_stac_guess_collection`

Guess the Collection id from an Asset id

Description

As the STAC catalogue contains a large number of records, your request may receive a timeout error. To prevent this it is best to narrow down your requests to specific collections. This function is a helper function that tries to guess the collection id from an asset id. Note that this method is not highly reliable, and it is always best to manually provide a collection id to a request.

Usage

```
dse_stac_guess_collection(asset_id)
```

Arguments

`asset_id`	An asset identifier name, used to guess its parent collection id.

Value

A `character` string with a guessed collection id. Or `NA` in case it cannot make a guess.

See Also

Other stac: `dse_stac_client()`, `dse_stac_collections()`, `dse_stac_download()`, `dse_stac_get_uri()`, `dse_stac_queryables()`, `dse_stac_search_request()`

Examples

```
dse_stac_guess_collection(
  "S2A_MSIL1C_20260109T132741_N0511_R024_T39XVL_20260109T142148")
```

dse_stac_queryables *Get Queryables for a STAC Collection*

Description

When searching through a collection with dse_stac_search_request(), it can be helpful to know which elements can be used to filter the search results (using dplyr::filter()). Calling dse_stac_queryables() tells you which aspects are available for querying and expected formats.

Usage

```
dse_stac_queryables(collection, ...)
```

Arguments

collection	Name of the collection for which to get the queryables.
...	Ignored

Value

Returns a named list with information about elements that can be used to query the collection

See Also

Other stac: dse_stac_client(), dse_stac_collections(), dse_stac_download(), dse_stac_get_uri(), dse_stac_guess_collection(), dse_stac_search_request()

Examples

```
if (interactive()) {
  dse_stac_queryables("sentinel-1-grd")
}
```

dse_stac_search_request *Create a Request for a STAC Search in the Data Space Ecosystem*

Description

In order to perform a search using the STAC API, you first need to create a request using dse_stac_search_request(). This creates a httr2::request() to which tidy verbs ?tidy_verbs can be applied (e.g., dplyr::select(), dplyr::filter() and dplyr::arrange(). Results are retrieved by calling dplyr::collect() on the request.

Usage

```
dse_stac_search_request(collections, ids, ...)
```

Arguments

collections	Restrict the search to the collections listed here.
ids	Restrict the search to ids listed here.
...	Arguments appended to search filter request body.

Details

If you prefer a graphical user interface, you can alternatively use the STAC web browser.

Value

Returns a data.frame with search results.

See Also

Other stac: dse_stac_client(), dse_stac_collections(), dse_stac_download(), dse_stac_get_uri(), dse_stac_guess_collection(), dse_stac_queryables()

Examples

```
if (interactive()) {
  library(dplyr)
  library(sf)

  bbox <-
    sf::st_bbox(
      c(xmin = 5.261, ymin = 52.680, xmax = 5.319, ymax = 52.715),
      crs = 4326)

  dse_stac_search_request("sentinel-2-l1c") |>
    filter(`eo:cloud_cover` < 10) |>
    collect()

  dse_stac_search_request("sentinel-1-grd") |>
    filter(`sat:orbit_state` == "ascending") |>
    arrange("id") |>
    st_intersects(bbox) |>
    collect()
}
```

---

dse_usage *Get Data Space Account Information*

Description

In order to guarantee good performance for all users, the Sentinel Hub applies rate limiting. This policy enforces monthly quotas to your usage. To check your quota and current usage, you can call dse_usage() or dse_user_statistics().

Usage

```
dse_usage(..., token = dse_access_token())

dse_user_statistics(
  range = "DAYS-31",
  resolution = "DAILY",
  token = dse_access_token()
)
```

Arguments

...	Ignored
token	For authentication, many of the Data Space Ecosystem uses an access token. Either provide your access token, or obtain one automatically with dse_access_token() (default). Without a valid token you will likely get an "access denied" error.
range	Specify a time range for which to obtain user statistics. The API expects a string starting with a capitalised time unit ("DAYS", "HOURS"), followed by a dash ("-") and an integer value specifying the length of the period. Default is "DAYS-31".
resolution	Specifying a temporal resolution for the user statistics. should be one of "DAILY" (default), "MONTHLY", or "HOURLY".

Value

A data.frame with requested information for the user associated with the provided token.

See Also

dse_access_token()

Other authentication: dse_access_token(), dse_get_token_details(), dse_has_s3_secret(), dse_set_gdal_token()

Other authentication: dse_access_token(), dse_get_token_details(), dse_has_s3_secret(), dse_set_gdal_token()

Examples

```
if (interactive() && dse_has_client_info()) {
  dse_usage()
  dse_user_statistics()
}
```

req_perform *Perform a Request to Get a Response*

Description

A wrapper around `httr2::req_perform()`, which can also handle `odata_request` class objects. Check `httr2::req_perform()` for details.

Usage

```
req_perform(
  req,
  path = NULL,
  verbosity = NULL,
  mock = getOption("httr2_mock", NULL),
  error_call = rlang::current_env()
)
```

Arguments

`req`	Either a `httr2::request()` class object or an `odata_request` class object. The latter can be created with `dse_odata_products_request()` and `dse_odata_bursts_request()`.
`path`	Optionally, path to save body of the response. This is useful for large responses since it avoids storing the response in memory.
`verbosity`	How much information to print? This is a wrapper around `req_verbose()` that uses an integer to control verbosity: • `0`: no output • `1`: show headers • `2`: show headers and bodies • `3`: show headers, bodies, and curl status messages. Use `with_verbosity()` to control the verbosity of requests that you can't affect directly.
`mock`	A mocking function. If supplied, this function is called with the request. It should return either `NULL` (if it doesn't want to handle the request) or a response (if it does). See `with_mocked_responses()`/`local_mocked_responses()` for more details.
`error_call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

Returns a `httr2::response` class object

See Also

Other tidyverse: tidy_verbs

---

st_intersects *Filter OData and STAC Requests Using Geometries*

Description

Filters OData and STAC rows that intersect with spatial feature y.

Usage

```
st_intersects.odata_request(x, y, sparse = FALSE, ...)

st_intersects.stac_request(x, y, sparse = FALSE, ...)
```

Arguments

x	Either an odata_request class object, generated with dse_odata_products_request()/dse_odata_bu or a stac_request generated with dse_stac_search_request().
y	A spatial geometry of either class sf (see sf::st_as_sf()) or sfc (see sf::st_as_sfc()). It will always be transformed to WGS 84 projection (EPSG:4326).
sparse	Argument inherited from generic definition. Ignored in this context
...	Ignored

Value

Returns an object of the same class as x, with the geometry filter added to it.

Examples

```
if (interactive()) {
  bbox <-
    sf::st_bbox(
      c(xmin = 5.261, ymin = 52.680, xmax = 5.319, ymax = 52.715),
      crs = 4326) |>
      sf::st_as_sfc()

  dse_odata_products_request() |>
    dplyr::filter(
      `ContentDate/Start` > "2025-01-01") |>
    sf::st_intersects(bbox) |>
    dplyr::collect()

  dse_stac_search_request("sentinel-2-l1c") |>
    sf::st_intersects(bbox) |>
    dplyr::collect()
}
```

tidy_verbs *Tidy Verbs for OData, SentinelHub and STAC API Requests*

Description

Implementation of tidy generics for features supported any of OData, SentinelHub or STAC API requests. They can be called on objects any of the classes: `odata_request`, `sentinel_request` or `stac_request`. The first is produced by `dse_odata_products_request()` and `dse_odata_bursts_request()`; the second by `dse_sh_search_request()`; and the last by `dse_stac_search_request()`.

Usage

```
filter.odata_request(.data, ..., .by = NULL, .preserve = FALSE)

filter.sentinel_request(.data, ..., .by = NULL, .preserve = FALSE)

filter.stac_request(.data, ..., .by = NULL, .preserve = FALSE)

compute.odata_request(x, skip = 0L, ...)

collect.odata_request(x, skip = 0L, ...)

collect.sentinel_request(x, skip = 0L, ...)

collect.stac_request(x, ...)

arrange.odata_request(.data, ..., .by_group = FALSE)

arrange.stac_request(.data, ..., .by_group = FALSE)

slice_head.odata_request(.data, ..., n, prop, by = NULL)

slice_head.stac_request(.data, ..., n, prop, by = NULL)

slice_head.sentinel_request(.data, ..., n, prop, by = NULL)

select.odata_request(.data, ...)

select.stac_request(.data, ...)

select.sentinel_request(.data, ...)
```

Arguments

`.data`, `x`	An object of any of the following classes `odata_request`, `sentinel_request` or `stac_request`. These are produced by `dse_odata_products_request()`, `dse_odata_bursts_request()`, `dse_sh_search_request()` and `dse_stac_search_request()`

`...` Data masking expressions, or arguments passed to embedded functions

`skip` Number of rows to skip when collecting results. The APIs return a limited number rows. Specify the number of rows to skip in order to get results beyond the predefined limit.

`n` Maximum number of rows to return.

`by`, `.by`, `.by_group`, `.preserve`, `prop`
Arguments inherited from generic `dplyr` functions. Ignored in the current context as either grouping is not allowed for an OData API request or is otherwise not supported.

Details

These special request class objects use lazy evaluation. This means that functions are only evaluated after calling `dplyr::collect()` on a request.

Note that you should not call the functions exported in this package directly. Instead, call the generics as declared in the `dplyr` package. This is illustrated by the examples.

Slice Head:

In order to manage server traffic, the OData API never returns more than 20 rows. If you want to obtain results beyond the first 20 rows, you need to specify the `skip` argument.

The Sentinel and STAC API limits its results to the first 10 rows. You can expand that limit with `dplyr::slice_head()`. For STAC the number of rows is capped at 10,000 records. For SentinelHub this number is capped at 100.

Deviations:

Due to limitations posed by the OData API, some tidyverse verbs deviate from its tidy standards. Most notably:

- `dplyr::select()`: Cannot change the order of columns. It will only affect which columns are selected. Also, tidy selection helpers like `dplyr::any_of()` and `dplyr::all_of()` are NOT supported
- `dplyr::arrange()`: OData only allows to sort up to 32 columns. Adding more columns will produce a warning.
- Grouping is not supported
- Only tidy methods listed in the usage section are supported for the special request class objects. If you want to apply the full spectrum of tidyverse methods, call `dplyr::collect()` on the request class object first. That will return a normal `data.frame`, which can be manipulated further.

Value

All functions (except `collect()`) return a modified `stac_request`/`sentinel_request`/`odata_request` object, containing the lazy tidy operations. `collect()` will return a `data.frame()` yielding the result of the request.

See Also

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Other tidyverse: req_perform()

Examples

```
library(dplyr)
if (interactive()) {
  dse_odata_products_request() |>
    filter(contains(Name, "WRR")) |>
    select("Id", "Name") |>
    arrange(Id, desc(Name)) |>
    slice_head(n = 5) |>
    collect()

  dse_stac_search_request("sentinel-1-grd") |>
    filter(`sat:orbit_state` == "ascending") |>
    arrange("id") |>
    collect()

  if (dse_has_client_info()) {
    dse_sh_search_request(
      collection = "sentinel-2-l2a",
      bbox       = c(5.261, 52.680, 5.319, 52.715),
      datetime   = c("2025-01-01 UTC", "2025-01-31 UTC")
    ) |>
      filter(`eo:cloud_cover` <= 10) |>
      collect()
  }
}
```

Index