Sentinelsat Documentation

Release 0.12

Marcel Wille, Kersten Clauss

Aug 16, 2017

Contents

Sentinelsat makes searching, downloading and retrieving the metadata of Sentinel satellite images from the Copernicus Open Access Hub easy.

It offers an easy-to-use command line interface

```
sentinelsat -u <user> -p <password> -g <search_polygon.geojson> --sentinel 2 --cloud␣
↪30
```

and a powerful Python API.

```
from sentinelsat import SentinelAPI, read_geojson, geojson_to_wkt

api = SentinelAPI('user', 'password')
footprint = geojson_to_wkt(read_geojson('search_polygon.geojson'))
products = api.query(footprint,
                     producttype='SLC',
                     orbitdirection='ASCENDING')
api.download_all(products)
```

CHAPTER 1

Contents

Installation

Install `sentinelsat` through pip:

```
pip install sentinelsat
```

Tests

To run the tests on `sentinelsat`:

```
git clone https://github.com/sentinelsat/sentinelsat.git
cd sentinelsat
pip install -e .[test]
py.test -v
```

By default, prerecorded responses to Copernicus Open Access Hub queries are used to not be affected by its downtime. To allow the tests to run actual queries against Copernicus Open Access Hub set the environment variables

```
export SENTINEL_USER=<your scihub username>
export SENTINEL_PASSWORD=<your scihub password>
```

and add `--vcr disable` to `py.test` arguments. To update the recordings use either `--vcr record_new` or `--vcr reset`.

Supported Python versions

Sentinelsat has been tested with Python versions 2.7 and 3.4+. Earlier Python 3 versions are expected to work as well as long as the dependencies are fulfilled.

Optional dependencies

The convenience functions `to_dataframe()` and `to_geodataframe()` require `pandas` and/or `geopandas` to be present.

Command Line Interface

Sentinelsat provides a CLI `sentinelsat` to query and download multiple or single images.

Quickstart

A basic search query consists of a search polygon as well as the username and password to access the Copernicus Open Access Hub.

```
sentinelsat -u <user> -p <password> -g <geojson>
```

Search areas are provided as GeoJSON polygons, which can be created with QGIS or geojson.io. If you do not specify a start and end date only products published in the last 24 hours will be queried.

Start and end dates refer to the acquisition date given by the beginPosition of the products, i.e. the start of the acquisition time.

Sentinel-1

Search and download all Sentinel-1 scenes of type SLC over a search polygon, in descending orbit for the year 2015.

```
sentinelsat -u <user> -p <password> -g <search_polygon.geojson> -s 20150101 -e␣
↪20151231 -d \
--producttype SLC -q "orbitdirection=Descending" \
--url "https://scihub.copernicus.eu/dhus"
```

Download a single Sentinel-1 GRDH scene covering Santa Claus Village in Finland on Christmas Eve 2015.

```
sentinelsat -u <user> -p <password> -d --md5 --uuid a9048d1d-fea6-4df8-bedd-
↪7bcb212be12e
```

or by using its filename

```
sentinelsat -u <user> -p <password> -d --md5 --name S1A_EW_GRDM_1SDH_20151224T154142_
↪20151224T154207_009186_00D3B0_C71E
```

Sentinel-2

Search and download Sentinel-2 scenes for January 2016 with a maximum cloud cover of 40%.

```
sentinelsat -u <user> -p <password> -g <search_polygon.geojson> -s 20160101 -e␣
↪20160131 --sentinel 2 --cloud 40 -d
```

Download all Sentinel-2 scenes published in the last 24 hours.

```
sentinelsat -u <user> -p <password> -g <search_polygon.geojson> --sentinel 2 -d
```

sentinelsat

```
sentinelsat -u <user> -p <password> [OPTIONS]
```

Options:

-u	–user	TEXT	Username [required]
-p	–password	TEXT	Password [required]
	–url	TEXT	Define another API URL. Default URL is ‘https://scihub.copernicus.eu/apihub/‘.
-s	–start	TEXT	Start date of the query in the format YYYYMMDD.
-e	–end	TEXT	End date of the query in the format YYYYMMDD.
-g	–geometry	PATH	Search area geometry as GeoJSON file.
	–uuid	TEXT	Select a specific product UUID instead of a query. Multiple UUIDs can separated by commas.
	–name	TEXT	Select specific product(s) by filename. Supports wildcards.
	–sentinel		Limit search to a Sentinel satellite (constellation).
	–instrument		Limit search to a specific instrument on a Sentinel satellite.
	–producttype		Limit search to a Sentinel product type.
-c	–cloud	INT	Maximum cloud cover in percent. (requires –sentinel to be 2 or 3)
-o	–order-by	TEXT	Comma-separated list of keywords to order the result by. Prefix ‘-‘ for descending order.
-l	–limit	INT	Maximum number of results to return. Defaults to no limit.
-d	–download		Download all results of the query.
	–path	PATH	Set the path where the files will be saved.
-q	–query	TEXT	Extra search keywords you want to use in the query. Separate keywords with comma. Example: ‘producttype=GRD,polarisationmode=HH’.
-f	–footprints		Create geojson file search_footprints.geojson with footprints of the query result.
	–md5		Verify the MD5 checksum and write corrupt product ids and filenames to corrupt_scenes.txt.
	–version		Show version number and exit.
	–help		Show help message and exit.

ESA maintains a list of valid search keywords that can be used with `--query`.

The options `--sentinel`, `--instrument` and `--producttype` are mutually exclusive and follow a hierarchy from most specific to least specific, i.e. `--producttype` > `--instrument` > `--sentinel`. Only the most specific option will be included in the search when multiple ones are given.

Searching by name supports wildcards, such as `S1A_IW*20151224*` to find all Sentinel-1 A scenes from 24th of December 2015 without restricting the result to a search area.

Python API

Quickstart

```
# connect to the API
from sentinelsat import SentinelAPI, read_geojson, geojson_to_wkt
api = SentinelAPI('user', 'password', 'https://scihub.copernicus.eu/dhus')

# download single scene by known product id
api.download(<product_id>)

# search by polygon, time, and SciHub query keywords
footprint = geojson_to_wkt(read_geojson('map.geojson'))
products = api.query(footprint,
                     '20151219', date(2015, 12, 29),
                     platformname = 'Sentinel-2',
                     cloudcoverpercentage = '[0 TO 30]')

# download all results from the search
api.download_all(products)

# GeoJSON FeatureCollection containing footprints and metadata of the scenes
api.to_geojson(products)

# GeoPandas GeoDataFrame with the metadata of the scenes and the footprints as␣
↪geometries
api.to_geopandas(products)

# Get basic information about the product: its title, file size, MD5 sum, date,␣
↪footprint and
# its download url
api.get_product_odata(<product_id>)

# Get the product's full metadata available on the server
api.get_product_odata(<product_id>, full=True)
```

Valid search query keywords can be found at the Copernicus Open Access Hub documentation.

Sorting & Filtering

In addition to the search query keywords sentinelsat allows filtering and sorting of search results before download. To simplify these operations sentinelsat offers the convenience functions `to_geojson()`, `to_dataframe()` and `to_geodataframe()` which return the search results as a GeoJSON object, Pandas DataFrame or a GeoPandas GeoDataFrame, respectively. `to_dataframe()` and `to_geodataframe()` require `pandas` and `geopandas` to be installed, respectively.

In this example we query Sentinel-2 scenes over a location and convert the query results to a Pandas DataFrame. The DataFrame is then sorted by cloud cover and ingestion date. We limit the query to first 5 results within our timespan and download them, starting with the least cloudy scene. Filtering can be done with all data types, as long as you pass the `id` to the download function.

```
# connect to the API
from sentinelsat import SentinelAPI, read_geojson, geojson_to_wkt

api = SentinelAPI('user', 'password', 'https://scihub.copernicus.eu/dhus')
```

```
# search by polygon, time, and SciHub query keywords
footprint = geojson_to_wkt(read_geojson('map.geojson'))
products = api.query(footprint,
                     '20151219', date(2015, 12, 29),
                     platformname = 'Sentinel-2')

# convert to Pandas DataFrame
products_df = api.to_dataframe(products)

# sort and limit to first 5 sorted products
products_df_sorted = products_df.sort_values(['cloudcoverpercentage', 'ingestiondate
↪'], ascending=[True, True])
products_df_sorted = products_df_sorted.head(5)

# download sorted and reduced products
api.download_all(products_df_sorted['id'])
```

Getting Product Metadata

Sentinelsat provides two methods for retrieving product metadata from the server, one for each API offered by the Copernicus Open Access Hub:

- `query()` for OpenSearch (Solr), which supports filtering products by their attributes and returns metadata for all matched products at once.
- `get_product_odata()` for OData, which can be queried one product at a time but provides the full metadata available for each product, as well as information about the product file such as the file size and checksum, which are not available from OpenSearch.

Both methods return a dictionary containing the metadata items. More specifically, `query()` returns a dictionary with an entry for each returned product with its ID as the key and the attributes' dictionary as the value.

All of the attributes returned by the OpenSearch API have a corresponding but differently named attribute in the OData's full metadata response. See the DataHubSystem's metadata definition files to find the exact mapping between them (OpenSearch attributes have a `<solrField>` tag added): - Sentinel-1 attributes - Sentinel-2 attributes - Sentinel-3 attributes

OpenSearch example

```
>>> api.query(date=('NOW-8HOURS', 'NOW'), producttype='SLC')
OrderedDict([('04548172-c64a-418f-8e83-7a4d148adf1e',
              {'acquisitiontype': 'NOMINAL',
               'beginposition': datetime.datetime(2017, 4, 25, 15, 56, 12, 814000),
               'endposition': datetime.datetime(2017, 4, 25, 15, 56, 39, 758000),
               'filename': 'S1A_IW_SLC__1SDV_20170425T155612_20170425T155639_016302_
↪01AF91_46FF.SAFE',
               'footprint': 'POLYGON ((34.322010 0.401648,36.540989 0.876987,36.
↪884121 -0.747357,34.664474 -1.227940,34.322010 0.401648))',
               'format': 'SAFE',
               'gmlfootprint': '<gml:Polygon srsName="http://www.opengis.net/gml/srs/
↪epsg.xml#4326" xmlns:gml="http://www.opengis.net/gml">\n   <gml:outerBoundaryIs>\n ␣
↪    <gml:LinearRing>\n         <gml:coordinates>0.401648,34.322010 0.876987,36.
↪540989 -0.747357,36.884121 -1.227940,34.664474 0.401648,34.322010</gml:coordinates>
↪\n      </gml:LinearRing>\n   </gml:outerBoundaryIs>\n</gml:Polygon>',
               'identifier': 'S1A_IW_SLC__1SDV_20170425T155612_20170425T155639_016302_
↪01AF91_46FF',
               'ingestiondate': datetime.datetime(2017, 4, 25, 19, 23, 45, 956000),
               'instrumentname': 'Synthetic Aperture Radar (C-band)',
               'instrumentshortname': 'SAR-C SAR',
               'lastorbitnumber': 16302,
               'lastrelativeorbitnumber': 130,
               'link': "https://scihub.copernicus.eu/apihub/odata/v1/Products(
↪'04548172-c64a-418f-8e83-7a4d148adf1e')/$value",
               'link_alternative': "https://scihub.copernicus.eu/apihub/odata/v1/
↪Products('04548172-c64a-418f-8e83-7a4d148adf1e')/",
               'link_icon': "https://scihub.copernicus.eu/apihub/odata/v1/Products(
↪'04548172-c64a-418f-8e83-7a4d148adf1e')/Products('Quicklook')/$value",
               'missiondatatakeid': 110481,
               'orbitdirection': 'ASCENDING',
               'orbitnumber': 16302,
               'platformidentifier': '2014-016A',
               'platformname': 'Sentinel-1',
               'polarisationmode': 'VV VH',
               'productclass': 'S',
               'producttype': 'SLC',
               'relativeorbitnumber': 130,
               'sensoroperationalmode': 'IW',
               'size': '7.1 GB',
               'slicenumber': 8,
               'status': 'ARCHIVED',
               'summary': 'Date: 2017-04-25T15:56:12.814Z, Instrument: SAR-C SAR,␣
↪Mode: VV VH, Satellite: Sentinel-1, Size: 7.1 GB',
               'swathidentifier': 'IW1 IW2 IW3',
               'title': 'S1A_IW_SLC__1SDV_20170425T155612_20170425T155639_016302_
↪01AF91_46FF',
               'uuid': '04548172-c64a-418f-8e83-7a4d148adf1e'}),
...
```

OData example

Only the most basic information available from the OData API is returned by default, if `full=True` is not set. The full metadata query response is quite large and not always nrequired, so it is not requested by default.

```
>>> api.get_product_odata('04548172-c64a-418f-8e83-7a4d148adf1e')
{'date': datetime.datetime(2017, 4, 25, 15, 56, 12, 814000),
 'footprint': 'POLYGON((34.322010 0.401648,36.540989 0.876987,36.884121 -0.747357,34.
↪664474 -1.227940,34.322010 0.401648))',
 'id': '04548172-c64a-418f-8e83-7a4d148adf1e',
 'md5': 'E5855D1C974171D33EE4BC08B9D221AE',
 'size': 4633501134,
 'title': 'S1A_IW_SLC__1SDV_20170425T155612_20170425T155639_016302_01AF91_46FF',
 'url': "https://scihub.copernicus.eu/apihub/odata/v1/Products('04548172-c64a-418f-
↪8e83-7a4d148adf1e')/$value"}
```

With `full=True` we receive the full metadata available for the product.

```
>>> api.get_product_odata('04548172-c64a-418f-8e83-7a4d148adf1e', full=True)
{'Acquisition Type': 'NOMINAL',
 'Carrier rocket': 'Soyuz',
 'Cycle number': 107,
 'Date': datetime.datetime(2017, 4, 25, 15, 56, 12, 814000),
 'Filename': 'S1A_IW_SLC__1SDV_20170425T155612_20170425T155639_016302_01AF91_46FF.SAFE
↪',
 'Footprint': '<gml:Polygon srsName="http://www.opengis.net/gml/srs/epsg.xml#4326"␣
 ↪xmlns:gml="http://www.opengis.net/gml">\n   <gml:outerBoundaryIs>\n
 ↪<gml:LinearRing>\n         <gml:coordinates>0.401648,34.322010 0.876987,36.540989 -
 ↪0.747357,36.884121 -1.227940,34.664474 0.401648,34.322010</gml:coordinates>\n
 ↪</gml:LinearRing>\n   </gml:outerBoundaryIs>\n</gml:Polygon>',
 'Format': 'SAFE',
 'Identifier': 'S1A_IW_SLC__1SDV_20170425T155612_20170425T155639_016302_01AF91_46FF',
 'Ingestion Date': datetime.datetime(2017, 4, 25, 19, 23, 45, 956000),
 'Instrument': 'SAR-C',
 'Instrument abbreviation': 'SAR-C SAR',
 'Instrument description': '<a target="_blank" href="https://sentinel.esa.int/web/
 ↪sentinel/missions/sentinel-1">https://sentinel.esa.int/web/sentinel/missions/
 ↪sentinel-1</a>',
 'Instrument description text': 'The SAR Antenna Subsystem (SAS) is developed and␣
 ↪build by AstriumGmbH. It is a large foldable planar phased array antenna, which␣
 ↪isformed by a centre panel and two antenna side wings. In deployedconfiguration the␣
 ↪antenna has an overall aperture of 12.3 x 0.84 m.The antenna provides a fast␣
 ↪electronic scanning capability inazimuth and elevation and is based on low loss and␣
 ↪highly stablewaveguide radiators build in carbon fibre technology, which arealready␣
 ↪successfully used by the TerraSAR-X radar imaging mission.The SAR Electronic␣
 ↪Subsystem (SES) is developed and build byAstrium Ltd. It provides all radar control,
 ↪ IF/ RF signalgeneration and receive data handling functions for the SARInstrument.␣
 ↪The fully redundant SES is based on a channelisedarchitecture with one transmit and␣
 ↪two receive chains, providing amodular approach to the generation and reception of␣
 ↪wide-bandsignals and the handling of multi-polarisation modes. One keyfeature is␣
 ↪the implementation of the Flexible Dynamic BlockAdaptive Quantisation (FD-BAQ) data␣
 ↪compression concept, whichallows an efficient use of on-board storage resources and␣
 ↪minimisesdownlink times.',
 'Instrument mode': 'IW',
 'Instrument name': 'Synthetic Aperture Radar (C-band)',
 'Instrument swath': 'IW1 IW2 IW3',
 'JTS footprint': 'POLYGON ((34.322010 0.401648,36.540989 0.876987,36.884121 -0.
 ↪747357,34.664474 -1.227940,34.322010 0.401648))',
 'Launch date': 'April 3rd, 2014',
 'Mission datatake id': 110481,
 'Mission type': 'Earth observation',
 'Mode': 'IW',
 'NSSDC identifier': '2014-016A',
 'Operator': 'European Space Agency',
 'Orbit number (start)': 16302,
 'Orbit number (stop)': 16302,
 'Pass direction': 'ASCENDING',
 'Phase identifier': 1,
 'Polarisation': 'VV VH',
 'Product class': 'S',
 'Product class description': 'SAR Standard L1 Product',
 'Product composition': 'Slice',
 'Product level': 'L1',
 'Product type': 'SLC',
 'Relative orbit (start)': 130,
 'Relative orbit (stop)': 130,
 'Satellite': 'Sentinel-1',
 'Satellite description': '<a target="_blank" href="https://sentinel.esa.int/web/
 ↪sentinel/missions/sentinel-1">https://sentinel.esa.int/web/sentinel/missions/
 ↪sentinel-1</a>',
 'Satellite name': 'Sentinel-1',
 'Satellite number': 'A',
 'Sensing start': datetime.datetime(2017, 4, 25, 15, 56, 12, 814000),
 'Sensing stop': datetime.datetime(2017, 4, 25, 15, 56, 39, 758000),
 'Size': '7.1 GB',
 'Slice number': 8,
 'Start relative orbit number': 130,
 'Status': 'ARCHIVED',
 'Stop relative orbit number': 130,
 'Timeliness Category': 'Fast-24h',
 'date': datetime.datetime(2017, 4, 25, 15, 56, 12, 814000),
 'footprint': 'POLYGON((34.322010 0.401648,36.540989 0.876987,36.884121 -0.747357,34.
↪664474 -1.227940,34.322010 0.401648))',
 'id': '04548172-c64a-418f-8e83-7a4d148adf1e',
 'md5': 'E5855D1C974171D33EE4BC08B9D221AE',
 'size': 4633501134,
 'title': 'S1A_IW_SLC__1SDV_20170425T155612_20170425T155639_016302_01AF91_46FF',
 'url': "https://scihub.copernicus.eu/apihub/odata/v1/Products('04548172-c64a-418f-
↪8e83-7a4d148adf1e')/$value"}
```

Logging

Sentinelsat logs to `sentinelsat` and the API to `sentinelsat.SentinelAPI`.

There is no predefined logging handler, so in order to have your script print the log messages, either use `logging.baseConfig`

```
import logging

logging.basicConfig(format='%(message)s', level='INFO')
```

or add a custom handler for `sentinelsat` (as implemented in `cli.py`)

```
import logging

logger = logging.getLogger('sentinelsat')
logger.setLevel('INFO')

h = logging.StreamHandler()
h.setLevel('INFO')
fmt = logging.Formatter('%(message)s')
h.setFormatter(fmt)
logger.addHandler(h)
```

API

class `sentinelsat.`**`SentinelAPI`**(*user*, *password*, *api_url='https://scihub.copernicus.eu/apihub/'*, *show_progressbars=True*)

Class to connect to Copernicus Open Access Hub, search and download imagery.

Parameters **user** : string

username for DataHub

password : string

password for DataHub

api_url : string, optional

URL of the DataHub defaults to 'https://scihub.copernicus.eu/apihub'

show_progressbars : bool

Whether progressbars should be shown or not, e.g. during download. Defaults to True.

Attributes

session	(requests.Session object) Session to connect to DataHub
api_url	(str) URL to the DataHub
page_size	(int) number of results per query page current value: 100 (maximum allowed on ApiHub)

Methods

`check_files`([paths, ids, directory, delete])	Verify the integrity of product files on disk.
`check_query_length`(query)	Determine whether a query to the OpenSearch API is too long.
`count`(*args, **kwargs)	Get the number of products matching a query.
`download`(id[, directory_path, checksum])	Download a product.
`download_all`(products[, directory_path, ...])	Download a list of products.
`format_query`([area, date, raw, area_relation])	Create OpenSearch API query string
`get_product_odata`(id[, full])	Access OData API to get info about a product.
`get_products_size`(products)	Return the total file size in GB of all products in the OpenSearch response
`query`([area, date, raw, area_relation, ...])	Query the OpenSearch API with the coordinates of an area, a date interval and any other search keywords accepted by the API.
`query_raw`(query[, order_by, limit, offset])	Do a full-text query on the OpenSearch API using the format specified in
`to_dataframe`(products)	Return the products from a query response as a Pandas DataFrame with the values in their appropriate Python types.
`to_geodataframe`(products)	Return the products from a query response as a GeoPandas GeoDataFrame with the values in their appropriate Python types.
`to_geojson`(products)	Return the products from a query response as a GeoJSON with the values in their appropriate Python types.

`check_files`(*paths=None*, *ids=None*, *directory=None*, *delete=False*)

Verify the integrity of product files on disk.

Integrity is checked by comparing the size and checksum of the file with the respective values on the server.

The input can be a list of products to check or a list of IDs and a directory.

In cases where multiple products with different IDs exist on the server for given product name, the file is considered to be correct if any of them matches the file size and checksum. A warning is logged in such situations.

The corrupt products' OData info is included in the return value to make it easier to re-download the products, if necessary.

Parameters **paths** : list[string]

List of product file paths.

ids : list[string]

List of product IDs.

directory : string

Directory where the files are located, if checking based on product IDs.

delete : bool

Whether to delete corrupt products. Defaults to False.

Returns dict[str, list[dict]]

A dictionary listing the invalid or missing files. The dictionary maps the corrupt file paths to a list of OData dictionaries of matching products on the server (as returned by `SentinelAPI.get_product_odata()`).

static **`check_query_length`**(*query*)

Determine whether a query to the OpenSearch API is too long.

The length of a query string is limited to approximately 3893 characters but any special characters (that is, not alphanumeric or -_.~) are counted twice towards that limit.

Parameters **query** : str

The query string

Returns float

Ratio of the query length to the maximum length

Notes

The query size limit arises from a limit on the length of the server's internal query, which looks like

http://localhost:30333//solr/dhus/select?q=... &wt=xslt&tr=opensearch_atom.xsl&dhusLongName=Sentinels+Scientific+Da
&dhusServer=https%3A%2F%2Fscihub.copernicus.eu%2Fapihub%2F&originalQuery=...
&rows=100&start=0&sort=ingestiondate+desc

This function will estimate the length of the "q" and "originalQuery" parameters to determine whether the query will fail. Their combined length can be at most about 7786 bytes.

`count`(**args*, ***kwargs*)

Get the number of products matching a query.

This is a significantly more efficient alternative to doing len(api.query()), which can take minutes to run for queries matching thousands of products.

Parameters **Identical to the parameters of api.query().**

Returns int

The number of products matching a query.

`download`(*id*, *directory_path='.'*, *checksum=False*)

Download a product.

Uses the filename on the server for the downloaded file, e.g. "S1A_EW_GRDH_1SDH_20141003T003840_20141003T003920_002658_002F54_4DD1.zip".

Incomplete downloads are continued and complete files are skipped.

Parameters **id** : string

UUID of the product, e.g. 'a8dd0cfd-613e-45ce-868c-d79177b916ed'

directory_path : string, optional

Where the file will be downloaded

checksum : bool, optional

If True, verify the downloaded file's integrity by checking its MD5 checksum. Throws InvalidChecksumError if the checksum does not match. Defaults to False.

Returns **product_info** : dict

Dictionary containing the product's info from get_product_info() as well as the path on disk.

Raises **InvalidChecksumError**

If the MD5 checksum does not match the checksum on the server.

`download_all`(*products*, *directory_path='.'*, *max_attempts=10*, *checksum=False*)

Download a list of products.

Takes a list of product IDs as input. This means that the return value of query() can be passed directly to this method.

File names on the server are used for the downloaded files, e.g. "S1A_EW_GRDH_1SDH_20141003T003840_20141003T003920_002658_002F54_4DD1.zip".

In case of interruptions or other exceptions, downloading will restart from where it left off. Downloading is attempted at most max_attempts times to avoid getting stuck with unrecoverable errors.

Parameters **products** : list

List of product IDs

directory_path : string

Directory where the downloaded files will be downloaded

max_attempts : int, optional

Number of allowed retries before giving up downloading a product. Defaults to 10.

Returns dict[string, dict]

A dictionary containing the return value from download() for each successfully downloaded product.

set[string]

The list of products that failed to download.

Other Parameters **See download().**

Raises **Raises the most recent downloading exception if all downloads failed.**

static `format_query`(*area=None*, *date=None*, *raw=None*, *area_relation='Intersects'*, ***keywords*)

Create OpenSearch API query string

`get_product_odata`(*id*, *full=False*)

Access OData API to get info about a product.

Returns a dict containing the id, title, size, md5sum, date, footprint and download url of the product. The date field corresponds to the Start ContentDate value.

If `full` is set to True, then the full, detailed metadata of the product is returned in addition to the above. For a mapping between the OpenSearch (Solr) and OData attribute names see the following definition files: https://github.com/SentinelDataHub/DataHubSystem/blob/master/addon/sentinel-1/src/main/resources/META-INF/sentinel-1.owl https://github.com/SentinelDataHub/DataHubSystem/blob/master/addon/sentinel-2/src/main/resources/META-INF/sentinel-2.owl https://github.com/SentinelDataHub/DataHubSystem/blob/master/addon/sentinel-3/src/main/resources/META-INF/sentinel-3.owl

Parameters **id** : string

The ID of the product to query

full : bool

Whether to get the full metadata for the Product

Returns dict[str, Any]

A dictionary with an item for each metadata attribute

static `get_products_size`(*products*)

Return the total file size in GB of all products in the OpenSearch response

`query`(*area=None*, *date=None*, *raw=None*, *area_relation='Intersects'*, *order_by=None*, *limit=None*, *offset=0*, ***keywords*)

Query the OpenSearch API with the coordinates of an area, a date interval and any other search keywords accepted by the API.

Parameters **area** : str, optional

The area of interest formatted as a Well-Known Text string.

date : tuple of (str or datetime) or str, optional

A time interval filter based on the Sensing Start Time of the products. Expects a tuple of (start, end), e.g. ("NOW-1DAY", "NOW"). The timestamps can be either a Python datetime or a string in one of the following formats:

- yyyyMMdd
- yyyy-MM-ddThh:mm:ss.SSSZ (ISO-8601)
- yyyy-MM-ddThh:mm:ssZ
- NOW
- NOW-<n>DAY(S) (or HOUR(S), MONTH(S), etc.)
- NOW+<n>DAY(S)
- yyyy-MM-ddThh:mm:ssZ-<n>DAY(S)
- NOW/DAY (or HOUR, MONTH etc.) - rounds the value to the given unit

Alternatively, an already fully formatted string such as "[NOW-1DAY TO NOW]" can be used as well.

raw : str, optional

Additional query text that will be appended to the query.

area_relation : {'Intersection', 'Contains', 'IsWithin'}, optional

What relation to use for testing the AOI. Case insensitive.

- Intersects: true if the AOI and the footprint intersect (default)
- Contains: true if the AOI is inside the footprint

- IsWithin: true if the footprint is inside the AOI

order_by: str, optional

A comma-separated list of fields to order by (on server side). Prefix the field name by '+' or '-' to sort in ascending or descending order, respectively. Ascending order is used, if prefix is omitted. Example: "cloudcoverpercentage, -beginposition".

limit: int, optional

Maximum number of products returned. Defaults to no limit.

offset: int, optional

The number of results to skip. Defaults to 0.

Returns dict[string, dict]

Products returned by the query as a dictionary with the product ID as the key and the product's attributes (a dictionary) as the value.

Other Parameters Additional keywords can be used to specify other query parameters,

e.g. relativeorbitnumber=70.

See https://scihub.copernicus.eu/twiki/do/view/SciHubUserGuide/3FullTextSearch

for a full list.

Range values can be passed as two-element tuples, e.g. cloudcoverpercentage=(0, 30).

The time interval formats accepted by the ''date'' parameter can also be used with

any other parameters that expect time intervals (that is: 'beginposition', 'endposition',

'date', 'creationdate', and 'ingestiondate').

`query_raw`(*query*, *order_by=None*, *limit=None*, *offset=0*)

Do a full-text query on the OpenSearch API using the format specified in https://scihub.copernicus.eu/twiki/do/view/SciHubUserGuide/3FullTextSearch

DEPRECATED: use query(raw=...) instead. This method will be removed in the next major release.

Parameters query : str

The query string.

order_by: str, optional

A comma-separated list of fields to order by (on server side). Prefix the field name by '+' or '-' to sort in ascending or descending order, respectively. Ascending order is used, if prefix is omitted. Example: "cloudcoverpercentage, -beginposition".

limit: int, optional

Maximum number of products returned. Defaults to no limit.

offset: int, optional

The number of results to skip. Defaults to 0.

Returns dict[string, dict]

Products returned by the query as a dictionary with the product ID as the key and the product's attributes (a dictionary) as the value.

static **`to_dataframe`**(*products*)
Return the products from a query response as a Pandas DataFrame with the values in their appropriate Python types.

static **`to_geodataframe`**(*products*)
Return the products from a query response as a GeoPandas GeoDataFrame with the values in their appropriate Python types.

static **`to_geojson`**(*products*)
Return the products from a query response as a GeoJSON with the values in their appropriate Python types.

`sentinelsat.`**`read_geojson`**(*geojson_file*)
Read a GeoJSON file into a GeoJSON object.

`sentinelsat.`**`geojson_to_wkt`**(*geojson_obj*, *feature_number=0*, *decimals=4*)
Convert a GeoJSON object to Well-Known Text. Intended for use with OpenSearch queries.

In case of FeatureCollection, only one of the features is used (the first by default). 3D points are converted to 2D.

Parameters **geojson_obj** : dict

a GeoJSON object

feature_number : int, optional

Feature to extract polygon from (in case of MultiPolygon FeatureCollection), defaults to first Feature

decimals : int, optional

Number of decimal figures after point to round coordinate to. Defaults to 4 (about 10 meters).

Returns polygon coordinates

string of comma separated coordinate tuples (lon, lat) to be used by SentinelAPI

Exceptions

exception `sentinelsat.`**`SentinelAPIError`**(*msg=None*, *response=None*)
Invalid responses from DataHub.

exception `sentinelsat.`**`InvalidChecksumError`**
MD5 checksum of a local file does not match the one from the server.

Change Log

All notable changes to `sentinelsat` will be listed here.

[master] – 2017-XX-XX

Added

Changed

Deprecated

[0.12.0] – 2017-08-10

Added

- Option to change the type of spatial relation for the AOI in `query()`. The choices are 'Interesects', 'Contains' and 'IsWithin'.
- `order_by` option to `query()` which controls the fields by which the products are sorted on the server side before being returned. `-o/--order-by` on the CLI.
- `limit` the number of products returned by `query()` and to set the number of products to skip via `offset`. `-l/--limit` on the CLI.
- Added `raw` parameter to `query()` to append any additional raw query string to the query.
- Query parameters that take intervals as values can now be passed a tuple of the interval range values.
- Date validation and parsing has been extended to all date-type parameters in queries, such as 'ingestiondate'.
- Added `count()` which quickly returns the number of products matching a query on the server without retrieving the full response.
- Method `check_query_length` to check if a query will fail because of being excessively long.
- Option to adjust the number of decimal figures in the coordinates of the WKT string returned by `geojson_to_wkt()`.
- CLI option to query by UUID (`--uuid`) or filename (`--name`).
- A more informative error message is shown if a too long query string was likely the cause of the query failing on the server side. This can be useful if the WKT string length would cause the query to fail otherwise.
- Progressbars can be disabled by setting `show_progressbars` to `False`. Progressbars may be customized by overriding the `_tqdm()` method.
- Contribution guidelines.
- Tests for validity of documentation and RST files.

Changed

- Merged CLI subcommands `sentinel search` and `sentinel download` into `sentinelsat`.
- CLI uses keywords instead of positional arguments, i.e. `--user <username>`.
- `initial_date` and `end_date` parameters in `query()` have been replaced with a single `date` parameter that takes a tuple of start and end dates as input.
- Files being downloaded now include an '.incomplete' suffix in their name until the download is finished.
- Removed `check_existing` option from `download()` and `download_all()`. Similar functionality has been provided in the new `check_files()` function.
- `format_query_date` has been changed into a public function.
- Added a progressbar to long-running queries.
- Tests can now be run from any directory rather than the repository root.
- Made the query string slightly more compact by getting rid of unnecessary 'AND' operators, spaces and parentheses.

- Reduced the size of the VCR.py cassettes used in unit tests.
- changed license from AGPLv3 to GPLv3+

Deprecated

- `query_raw()` has been merged with `query()` and is deprecated. Use `query(raw=...)` instead.

Fixed

- Show the correct progress value in the download progressbar when continuing from an incomplete file. (Thanks @gbaier!)
- Added a workaround for a server-side bug when plus symbols are used in a query.

[0.11] – 2017-06-01

Changed

- Replace `pycurl` dependency with `requests`. This makes installation significantly easier. (#117)
- An exception is raised in `download_all()` if all downloads failed.
- Change 'Sentinels Scientific Datahub' to 'Copernicus Open Access Hub' (#100)
- Renamed `py.test` option `--vcr reset_all` to `--vcr reset` to better reflect its true behavior.

[0.10] – 2017-05-30

Added

- GeoJSON footprints are allowed to contain just a single geometry instead of a feature collection. Any geometry type that has a WKT equivalent is supported (rather than only Polygons).
- `get_product_odata()` can be used to get the full metadata information available for a product if `full=True` is set.
- Added `query_raw()` that takes full text search string as input and returns a parsed dictionary just like the updated `query()` method.
- CLI: `--sentinel=<int>` option to select satellite (constellation)

Changed

- `SentinelAPI`, etc. can be directly imported from `sentinelsat` rather than `sentinelsat.sentinel`.
- `query()` changes:
  - The `area` argument expects a WKT string as input instead of a coordinate string. (Issue #101)
  - Date arguments can be disabled by setting them to `None` and their values are validated on the client side. (Issue #101)
  - The return value has been changed to a dict of dicts of parsed metadata values. One entry per product with the product ID as the key.

- `download_all()` expects a list of product IDs as input. This is compatible with the output of `query()`.
- `get_coordinates()` has been replaced with functions `read_geojson()` and `geojson_to_wkt()`. (Issue #101)
- Use more compact and descriptive error messages from the response headers, if available.

Deprecated

- CLI: `--sentinel1` and `--sentinel2` will be removed with the next major release

Removed

- `to_dict()` has been removed since it is no longer required.
- `load_query()` has been made private (renamed to `_load_query()`).

Fixed

- Fixed invalid GeoJSON output in both the CLI and API. (Issue #104)
- Fixed broken reporting of failed downloads in the CLI. (Issue #88)
- Attempting to download a product with an invalid ID no longer creates an infinite loop and a more informative error message is displayed in the CLI.

[0.9.1] – 2017-03-06

Added

- `--version` option to command line utilities
- install requirements for building the documentation
- documentation of sorting with `to_*` convenience functions

[0.9] – 2017-02-26

Added

- Added `to_dict`, `to_dataframe` and `to_geodataframe` which convert the response content to respective types. The pandas, geopandas and shapely dependencies are not installed by default.

Changed

- `--footprints` now includes all returned product properties in the output.
- `KeyError('No results returned.')` is no longer returned for zero returned products in a response.
- Renamed `get_footprint` to `to_geojson` and `get_product_info` to `get_product_odata`.
- Added underscore to methods and functions that are not expected to be used outside the package.
- Instance variables `url` and `content` have been removed, `last_query` and `last_status_code` have been made private.

[0.8.1] – 2017-02-05

Added

- added a changelog

Changed

- use logging instead of print

Fixed

- docs represent new `query` and `download_all` behaviour

[0.8] – 2017-01-27

Added

- options to create new, reset or ignore vcr cassettes for testing

Changed

- `query` now returns a list of search results
- `download_all` requires the list of search results as an argument

Removed

- `SentinelAPI` does not save query results as class attributes

[0.7.4] – 2017-01-14

Added

- Travis tests for Python 3.6

[0.7.3] – 2016-12-09

Changed

- changed `SentinelAPI` `max_rows` attribute to `page_size` to better reflect pagination
- tests use `vcrpy` cassettes

Fixed

- support GeoJSON polygons with optional (third) z-coordinate

[0.7.1] – 2016-10-28

Added

- pagination support for query results

Changed

- number of query results per page set to 100

[0.6.5] – 2016-06-22

Added

- support for large queries

Changed

- Removed redundant information from Readme that is also present on Readthedocs

[0.6.4] – 2016-04-06-03

Changed

- `initial_date` / `--start` changed from ingestion to acquisition date

[0.6.1] – 2016-04-22

Added

- Sphinx documentation setup with autodoc and numpydoc
- Redthedocs.org integration

[0.5.5] – 2016-01-13

Added

- Sentinel-2 support

[0.5.1] – 2015-12-18

Added

- Travis added as continuous integration service for automated testing

[0.5] – 2015-12-09

Added

- validate downloaded products with their MD5 checksums

[0.4.3] – 2015-11-23

Added

- option to select a different dhus api `--url`

Changed

- `https://scihub.esa.int/apihub/` as standard url

[0.4] – 2015-09-28

Added

- method to manually select the CA certificate bundle
- function to return footprints of the queried Sentinel scenes

Fixed

- CA-certificate SSL errors

[0.3] – 2015-06-10

Added

- `--query` parameter to use extra search keywords in the cli

[0.1] – 2015-06-05

- first release

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index