Sentinelsat Documentation

Release 1.2.1

Marcel Wille, Kersten Clauss

Mar 10, 2023

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> --location Berlin --sentinel 2 --cloud 30 --start␣
↪NOW-1MONTH
```

and a powerful Python API.

```python
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',
                     limit=10)
api.download_all(products)
```

CHAPTER 1

Installation

Install `sentinelsat` through pip:

```
pip install sentinelsat
```

1.1 Tests

To run the tests on `sentinelsat`:

```
git clone https://github.com/sentinelsat/sentinelsat.git
cd sentinelsat
pip install -e .[dev]
pytest -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 DHUS_USER=<your scihub username>
export DHUS_PASSWORD=<your scihub password>
```

and add `--disable-vcr` to `pytest` arguments. To update the recordings use `--vcr-record` with `once`, `new_episodes` or `all`. See vcrpy docs for details. There are two minor issues to keep in mind when recording unit tests VCRs.

1. Between calls a formerly offline product can become available, if the previous call triggered its LTA retrieval.

2. dhus and apihub have different md5 hashes for products with the same UUID.

1.2 Supported Python versions

Sentinelsat has been tested with Python versions 3.5+. Earlier Python 3 versions are expected to work as well, as long as the dependencies are fulfilled.

1.3 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.

2.1 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 <search_polygon.geojson>
```

For convenience and added security, there are two ways you can store your credentials and omit them from the command line call. You can set username, password and DHuS URL as environment variables.

```
export DHUS_USER="<user>"
export DHUS_PASSWORD="<password>"
export DHUS_URL="<api_url>"
```

Alternatively, you can add them to a file *.netrc* in your user home directory.

```
machine apihub.copernicus.eu
login <user>
password <password>
```

Environment variables take precedence over *.netrc*. The above command then becomes

```
sentinelsat -g <search_polygon.geojson>
```

Search areas (i.e. search_polygon.geojson) are provided as GeoJSON files, 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.

2.1.1 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://apihub.copernicus.eu/apihub"
```

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

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

or by using its filename

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

2.1.2 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
```

2.1.3 Orbit products

Search precise orbit products ("AUX_POEORB") for Sentinel-1B for a specific date.

```
sentinelsat --gnss -s 20210201 -e 20210202 --producttype AUX_POEORB --query=
→"platformserialidentifier=1B"
```

2.2 Options

-u <username>, **--user** <username>
  Username. Required.

  Can also be set via the DHUS_USER environment variable or with a *.netrc file*.

-p <password>, **--password** <password>
  Password. Required.

  Can also be set with the DHUS_PASSWORD environment variable or with a *.netrc file*.

--url <api_url>
  Define another API URL. Default is 'https://apihub.copernicus.eu/apihub/'.

  Can also be set with the DHUS_URL environment variable.

-s <date>, **--start** <date>
    Start date of the query in the format YYYYMMDD or one of the other formats listed here.

-e <date>, **--end** <date>
    End date of the query in the format YYYYMMDD or one of the other formats listed here.

-g <file>, **--geometry** <file>
    Search area geometry as GeoJSON file.

--uuid
    Select a specific product UUID. Can be used more than once.

--name <name>
    Select specific product(s) by filename. Supports wildcards, such as `S1A_IW*20151224*` to find all Sentinel-1A scenes from 24th of December 2015 without restricting the result to a search area. Can be set more than once.

--sentinel <number>
    Limit search to a Sentinel satellite (constellation).

--instrument <instrument name>
    Limit search to a specific instrument on a Sentinel satellite.

--producttype <product type>
    Limit search to a Sentinel product type. List of valid product types can be found under *producttype* here.

-c <percent>, **--cloud** <percent>
    Maximum cloud cover in percent. (requires *--sentinel* to be 2 or 3)

-o <keywords>, **--order-by** <keywords>
    Comma-separated list of keywords to order the result by. Prefix with '-' for descending order.

-l <number>, **--limit** <number>
    Maximum number of results to return. Defaults to no limit.

-d, --download
    Download all results of the query.

--fail-fast
    Skip all other other downloads if one fails.

--path <directory>
    Set the directory where the files will be saved.

-q <query>, **--query** <query>
    Extra search keywords you want to use in the query. Repeated keywords get interpreted as an "or" expression.

    ESA maintains a list of valid search keywords that can be used.

    Example: *-q producttype=GRD -q polarisationmode=HH*.

-f, --footprints <path>
    Create a GeoJSON file at the provided path with footprints and metadata of the returned products. Set to '-' for stdout.

--include-pattern
    Glob pattern to filter files (within each product) to be downloaded.

--exclude-pattern
    Glob pattern to filter files (within each product) to be excluded from the downloaded.

--timeout <seconds>
    How long to wait for a DataHub response (in seconds, default 60 sec).

--gnss
> Query orbit products form the GNSS end-point (”https://scihub.copernicus.eu/gnss”).

--fmt `<format string>`
> Specify a custom format to print results. The format string shall be compatible with the Python "Format Specification Mini-Language".

> **Some common keywords for substitution are:** 'uuid', 'identifier', 'summary', 'link', 'size', 'platformname', 'producttype', 'beginposition', 'instrumentshortname', 'cloudcoverpercentage', 'orbitdirection', 'relative-orbitnumber', 'footprint'.

> For a complete set of available keywords see the "properties" output from a relevant query with `--footprints` – (and possibly `--limit 1`) appended.

> Default: "Product {uuid} - {summary}"

--info
> Display DHuS server information.

--version
> Show version number and exit.

--debug
> Print debug log messages.

-h, --help
> Show help message and exit.

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.

Also the `--include-pattern` and `--exclude-patter` options are mutually exclusive. If used together the CLI program exists with an error.

Python API

3.1 Quickstart

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

api = SentinelAPI('user', 'password', 'https://apihub.copernicus.eu/apihub')

# 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('/path/to/map.geojson'))
products = api.query(footprint,
                     date=('20151219', date(2015, 12, 29)),
                     platformname='Sentinel-2',
                     cloudcoverpercentage=(0, 30))

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

# convert to Pandas DataFrame
products_df = api.to_dataframe(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_geodataframe(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.

3.2 Authentication

The Copernicus Open Access Hub and probably most Data Hubs require authentication. You can provide your credentials with `SentinelAPI(<your username>, <your password>)`. Alternatively, you can use `SentinelAPI(None, None)` and enter your credentials in a file *.netrc* in your user home directory in the following form:

```
machine apihub.copernicus.eu
login <your username>
password <your password>
```

Either way, if you get an error 401 Unauthorized, your credentials were wrong or not yet active for the endpoint you are contacting.

3.3 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
from datetime import date

api = SentinelAPI('user', 'password', 'https://apihub.copernicus.eu/apihub')

# search by polygon, time, and SciHub query keywords
footprint = geojson_to_wkt(read_geojson('map.geojson'))
products = api.query(footprint,
                     date=('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.index)
```

3.4 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

3.4.1 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://apihub.copernicus.eu/apihub/odata/v1/Products(
→'04548172-c64a-418f-8e83-7a4d148adf1e')/$value",
                'link_alternative': "https://apihub.copernicus.eu/apihub/odata/v1/
→Products('04548172-c64a-418f-8e83-7a4d148adf1e')/",
                'link_icon': "https://apihub.copernicus.eu/apihub/odata/v1/Products(
→'04548172-c64a-418f-8e83-7a4d148adf1e')/Products('Quicklook')/$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'}),
...
```

3.4.2 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 required, 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://apihub.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://apihub.copernicus.eu/apihub/odata/v1/Products('04548172-c64a-418f-
→8e83-7a4d148adf1e')/$value"}
```

3.5 LTA-Products

Copernicus Open Access Hub no longer stores all products online for immediate retrieval. Offline products can be requested from the Long Term Archive (LTA) and should become available within 24 hours. Copernicus Open Access Hub's quota currently permits users to request an offline product every 30 minutes.

A product's availability can be checked with a regular OData query by evaluating the `Online` property value or by using the `is_online()` convenience method.

The retrieval of offline products from the LTA can be triggered using `trigger_offline_retrieval()`.

```python
product_info = api.get_product_odata(<product_id>)
is_online = product_info['Online']
# or
is_online = api.is_online(<product_id>)


if is_online:
    print(f'Product {<product_id>} is online. Starting download.')
    api.download(<product_id>)
else:
    print(f'Product {<product_id>} is not online.')
    api.trigger_offline_retrieval(<product_id>)
```

When trying to download an offline product with `download()` it will trigger its retrieval from the LTA and raise an `LTATriggered` exception.

Given a list of offline and online products, `download_all()` will download online products, while concurrently triggering the retrieval of offline products from the LTA in the background. `download_all()` terminates when all products have been retrieved from the LTA and downloaded. If you wish to avoid the possibly lengthy retrieval process of offline products, you can either set the `lta_timeout` parameter to a low value or filter the list of products based on the `is_online()` status beforehand.

3.6 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`

```python
import logging

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

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

```python
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)
```

3.7 Downloading parts of products

Both *download()* and *download_all()* include a `nodefilter` parameter that can be used to specify a subset of files within the product that should be downloaded, skipping the rest. The downloaded files will be written to disk as individual files instead of a single archive file. This functionality makes use of the node selection feature of the OData Web API.

```python
from sentinelsat import SentinelAPI, make_path_filter

# define the filter function to select files (to be excluded in this case)
path_filter = make_path_filter("*measurement/*", exclude=True)

# connect to the API
api = SentinelProductAPI("user", "password")

# download a single product excluding measurement files
api.download(<product_id>, nodefilter=path_filter)
```

Of course it also works for multiple products, too:

```python
# download a multiple products excluding measurement files
api.download_all(<products>, nodefilter=path_filter)
```

The example above downloads all files in each of the requested products only *excluding* (large) measurements files. This can be useful for analyses exclusively based on product annotations (including calibration annotations) or, e.g., to download the KML preview included in the product.

The file selection is implemented by specifying a *nodefilter* function that is called for each file (except for the manifest, which is always downloaded) in the requested products and returns `True` if the file have to be downloaded, `False` otherwise.

The *nodefilter* function has the following signature:

```python
def nodefilter(node_info: dict) -> bool:
    ...
```

The *node_info* parameter is a dictionary containing (at least) the following keys:

> **url** the URL to download the product file node

node_path the *path* within the product (e.g. "./preview/map-overlay.kml")

size the file size in bytes (int)

md5 the file's MD5 checksum

In the example above it has been used an helper function (`sentinelsat.products.make_path_filter()`), provided by the `sentinelsat.products` module, that generates *nodefilter* functions based on glob expressions applied to the *node_path* value.

The following code:

```
path_filter = make_path_filter("*measurement/*", exclude=True)
```

is more or less equivalent to:

```python
import fnmatch

def path_filter(node_info):
  return fnmatch.fnmatch(node_info["node_path"].lower(), pattern)
```

The `sentinelsat` product node API also provides:

- the `sentinelsat.products.make_size_filter()` helper to build filters based on the file size and

- the pre-build `sentinelsat.products.all_nodes_filter()` *nodefilter* function to download all files. This function can be used to download the entire Sentinel product as a directory instead of downloading a single zip archive.

Of course the user can write their own *nodefilter* functions if necessary.

3.8 More Examples

3.8.1 Search Sentinel-2 L1C by tile

To search for recent Sentinel-2 L1C imagery by MGRS tile, you can use the *tileid* parameter:

```python
from collections import OrderedDict
from sentinelsat import SentinelAPI

api = SentinelAPI('user', 'password')

tiles = ['33VUC', '33UUB']

query_kwargs = {
        'platformname': 'Sentinel-2',
        'producttype': 'S2MSI1C',
        'date': ('NOW-14DAYS', 'NOW')}

products = OrderedDict()
for tile in tiles:
    kw = query_kwargs.copy()
    kw['tileid'] = tile
    pp = api.query(**kw)
    products.update(pp)

api.download_all(products)
```

NB: Older products may not be found with the *tileid* parameter. On the Copernicus Open Access Hub, it seems to be available for most L1C products (product type S2MSI1C) from recent years, but this differs by region, too. To be on the safe side, combine the *tileid* search with a *filename* pattern search:

```
kw = query_kwargs.copy()
kw['raw'] = f'tileid:{tile} OR filename:*_T{tile}_*'
pp = api.query(**kw)
```

3.8.2 Download only some of the channels of a Sentinel-1 product

In some cases the user may be interested only to a specific sub-swath and/or polarization of Sentinel1 SLC products (e.g. for an interferometric analysis).

The following *nodefilter* function only downloads "HH" polarization measurement files for sub-swaths "EW1" and "EW2":

```
path_filter = make_path_filter("*s1?-ew[12]-slc-hh-*.tiff")
```

Considering that e.g. a Dual Pol Extended Wide Swath Sentinel-1 product includes 2 measurement files for each of the 5 sub-swath the above filter allows to reduce consistently the amount of data to be downloaded (form 10 to 2 TIFF files approx 700MB each).

```
api.download_all(<products>, nodefilter=path_filter)
```

API Reference

class SentinelAPI(*user*, *password*, *api_url='https://apihub.copernicus.eu/apihub/'*, *show_progressbars=True*, *timeout=60*)

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

> **Parameters**
>
> - **user** (`string`) – username for DataHub set to None to use ~/.netrc
> - **password** (`string`) – password for DataHub set to None to use ~/.netrc
> - **api_url** (`string, optional`) – URL of the DataHub defaults to 'https://apihub.copernicus.eu/apihub'
> - **show_progressbars** (`bool`) – Whether progressbars should be shown or not, e.g. during download. Defaults to True.
> - **timeout** (`float or tuple, default 60`) – How long to wait for DataHub response (in seconds). Tuple (connect, read) allowed. Set to None to wait indefinitely.

> **session**
>
> Session to connect to DataHub
>
> > **Type** requests.Session

> **api_url**
>
> URL to the DataHub
>
> > **Type** str

> **page_size**
>
> Number of results per query page. Current value: 100 (maximum allowed on ApiHub)
>
> > **Type** int

> **timeout**
>
> How long to wait for DataHub response (in seconds).
>
> > **Type** float or tuple

logger = <Logger sentinelsat.SentinelAPI (WARNING)>

concurrent_dl_limit
    Maximum number of concurrent downloads allowed by the server.

        Type int

concurrent_lta_trigger_limit
    Maximum number of concurrent Long Term Archive retrievals allowed.

        Type int

dl_retry_delay
    Number of seconds to wait between retrying of failed downloads.

        Type float, default 10

lta_retry_delay
    Number of seconds to wait between requests to the Long Term Archive.

        Type float, default 60

lta_timeout
    Maximum number of seconds to wait for triggered products to come online. Defaults to no timeout.

        Type float, optional

dl_limit_semaphore

lta_limit_semaphore

dhus_version

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** (`{'Intersects', '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.

- ****keywords** – 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)*. *None* can be used in range values for one-sided ranges, e.g. *orbitnumber=(16302, None)*. Ranges with no bounds (*orbitnumber=(None, None)*) will not be included in the query.

Multiple values for the same query parameter can be provided as sets and will be handled as logical OR, e.g. *orbitnumber={16302, 1206}*.

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').

> **Returns** 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.
>
> **Return type** dict[string, dict]

static format_query (*area=None*, *date=None*, *raw=None*, *area_relation='Intersects'*, ***keywords*)
Create a OpenSearch API query string.

count (*area=None*, *date=None*, *raw=None*, *area_relation='Intersects'*, ***keywords*)
Get the number of products matching a query.

Accepted parameters are identical to *SentinelAPI.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.

> **Returns** The number of products matching a query.
>
> **Return type** int

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

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.

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.

> **Parameters**
>
> - **id** (*string*) – The UUID of the product to query
>
> - **full** (*bool*) – Whether to get the full metadata for the Product. False by default.
>
> **Returns** A dictionary with an item for each metadata attribute
>
> **Return type** dict[str, Any]

Notes

For a full list of mappings 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

is_online(*id*)

Returns whether a product is online

> **Parameters id** (*string*) – UUID of the product, e.g. 'a8dd0cfd-613e-45ce-868c-d79177b916ed'
>
> **Returns** True if online, False if in LTA
>
> **Return type** bool

See also:

SentinelAPI.trigger_offline_retrieval()

download(*id*, *directory_path='.'*, *checksum=True*, *nodefilter=None*)

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, default True*) – If True, verify the downloaded file's integrity by checking its checksum. Throws InvalidChecksumError if the checksum does not match.
>
> - **nodefilter** (*callable, optional*) – The callable is used to select which files of each product will be downloaded. If None (the default), the full products will be downloaded. See *sentinelsat.products* for sample node filters.
>
> **Returns** **product_info** – Dictionary containing the product's info from get_product_odata() as well as the path on disk.
>
> **Return type** dict
>
> **Raises**

- `InvalidChecksumError` – If the MD5 checksum does not match the checksum on the server.

- `LTATriggered` – If the product has been archived and its retrieval was successfully triggered.

- `LTAError` – If the product has been archived and its retrieval failed.

trigger_offline_retrieval(*uuid*)

Triggers retrieval of an offline product from the Long Term Archive.

> **Parameters uuid** (*string*) – UUID of the product
>
> **Returns** True, if the product retrieval was successfully triggered. False, if the product is already online.
>
> **Return type** bool
>
> **Raises**
>
> - `LTAError` – If the request was not accepted due to exceeded user quota or server overload.
>
> - `ServerError` – If an unexpected response was received from server.
>
> - `UnauthorizedError` – If the provided credentials were invalid.

Notes

https://scihub.copernicus.eu/userguide/LongTermArchive

download_all(*products,* *directory_path='.',* *max_attempts=10,* *checksum=True,* *n_concurrent_dl=None, lta_retry_delay=None, fail_fast=False, nodefilter=None*)

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, default 10*) – Number of allowed retries before giving up downloading a product.
>
> - **checksum** (*bool, default True*) – If True, verify the downloaded files' integrity by checking its MD5 checksum. Throws InvalidChecksumError if the checksum does not match. Defaults to True.
>
> - **n_concurrent_dl** (*integer, optional*) – Number of concurrent downloads. Defaults to *SentinelAPI.concurrent_dl_limit*.
>
> - **lta_retry_delay** (*float, default 60*) – Number of seconds to wait between requests to the Long Term Archive.

- **fail_fast** (*bool, default False*) – if True, all other downloads are cancelled when one of the downloads fails.

- **nodefilter** (*callable, optional*) – The callable is used to select which files of each product will be downloaded. If None (the default), the full products will be downloaded. See *sentinelsat.products* for sample node filters.

Notes

By default, raises the most recent downloading exception if all downloads failed. If `fail_fast` is set to True, raises the encountered exception on the first failed download instead.

Returns

- *dict[string, dict]* – A dictionary containing the return value from download() for each successfully downloaded product.

- *dict[string, dict]* – A dictionary containing the product information for products successfully triggered for retrieval from the long term archive but not downloaded.

- *dict[string, dict]* – A dictionary containing the product information of products where either downloading or triggering failed. "exception" field with the exception info is included to the product info dict.

download_all_quicklooks (*products*, *directory_path='.'*, *n_concurrent_dl=4*)
Download quicklook for a list of products.

Takes a dict of product IDs: product data 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 images, e.g. "S2A_MSIL1C_20200924T104031_N0209_R008_T35WMV_20200926T135405.jpeg".

Parameters

- **products** (*dict*) – Dict of product IDs, product data

- **directory_path** (*string*) – Directory where the downloaded images will be downloaded

- **n_concurrent_dl** (*integer*) – Number of concurrent downloads

Returns

- *dict[string, dict]* – A dictionary containing the return value from download_quicklook() for each successfully downloaded quicklook

- *dict[string, dict]* – A dictionary containing the error of products where either quicklook was not available or it had an unexpected content type

download_quicklook (*id*, *directory_path='.'*)
Download a quicklook for a product.

Uses the filename on the server for the downloaded image name, e.g. "S1A_EW_GRDH_1SDH_20141003T003840_20141003T003920_002658_002F54_4DD1.jpeg".

Already downloaded images are skipped.

Parameters

- **id** (*string*) – UUID of the product, e.g. 'a8dd0cfd-613e-45ce-868c-d79177b916ed'

- **directory_path** (*string, optional*) – Where the image will be downloaded

> > **Returns quicklook_info** – Dictionary containing the quicklooks's response headers as well as the path on disk.
>
> > **Return type** dict

static get_products_size(*products*)

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

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 3898 characters but any special characters (that is, not alphanumeric or -_.*) will take up more space.
>
> > **Parameters query** (*str*) – The query string
>
> > **Returns** Ratio of the query length to the maximum length
>
> > **Return type** float

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** 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()*).
> >
> > **Return type** dict[str, list[dict]]

get_stream(*id*, ***kwargs*)

> Exposes requests response ready to stream product to e.g. S3.
>
> > **Parameters**
> >
> > - **id** (*string*) – UUID of the product, e.g. 'a8dd0cfd-613e-45ce-868c-d79177b916ed'
> > - ****kwargs** – Any additional parameters for requests.get()
> >
> > **Raises**
> >
> > - LTATriggered – If the product has been archived and its retrieval was successfully triggered.
> > - LTAError – If the product has been archived and its retrieval failed.

> **Returns** Opened response object
>
> **Return type** requests.Response

read_geojson(*geojson_file*)

> Read a GeoJSON file into a GeoJSON object.

geojson_to_wkt(*geojson_obj*, *decimals=4*)

> Convert a GeoJSON object to Well-Known Text. Intended for use with OpenSearch queries. 3D points are converted to 2D.
>
> > **Parameters**
> >
> > - **geojson_obj** (`dict`) – a GeoJSON object
> >
> > - **decimals** (`int, optional`) – Number of decimal figures after point to round coordinate to. Defaults to 4 (about 10 meters).
> >
> > **Returns** Well-Known Text string representation of the geometry
> >
> > **Return type** str

format_query_date(*in_date*)

> Format a date, datetime or a YYYYMMDD string input as YYYY-MM-DDThh:mm:ssZ or validate a date string as suitable for the full text search interface and return it.
>
> *None* will be converted to '*', meaning an unlimited date bound in date ranges.
>
> > **Parameters in_date** (`str or datetime or date or None`) – Date to be formatted
> >
> > **Returns** Formatted string
> >
> > **Return type** str
> >
> > **Raises** `ValueError` – If the input date type is incorrect or passed date string is invalid

placename_to_wkt(*place_name*)

> **Geocodes the place name to rectangular bounding extents using Nominatim API and** returns the corresponding 'ENVELOPE' form Well-Known-Text.
>
> > **Parameters place_name** (`str`) – the query to geocode
> >
> > **Raises** `ValueError` – If no matches to the place name were found.
> >
> > **Returns**
> >
> > - **wkt_envelope** (*str*) – Bounding box of the location as an 'ENVELOPE(minX, maxX, maxY, minY)' WKT string.
> >
> > - **info** (*Dict[str, any]*) – Matched location's metadata returned by Nominatim.

is_wkt(*possible_wkt*)

class SentinelProductsAPI(**args*, ***kwargs*)

> Bases: `sentinelsat.sentinel.SentinelAPI`
>
> Deprecated since version 1.1.0: SentinelProductsAPI functionality has been merged into SentinelAPI and Downloader and will be removed in a future release.

make_size_filter(*max_size*)

> Generate a nodefilter function to download only files below the specified maximum size.
>
> New in version 0.15.

---

make_path_filter(*pattern*, *exclude=False*)

> Generate a nodefilter function to download only files matching the specified pattern.

> > **Parameters**

> > > - **pattern** (*str*) – glob patter for files selection

> > > - **exclude** (*bool, optional*) – if set to True then files matching the specified pattern are excluded. Default False.

> > > - **versionadded:** (.) – 0.15:

all_nodes_filter(*node_info*)

> Node filter function to download all files.

> This function can be used to download a Sentinel product as a directory instead of downloading a single zip archive.

> New in version 0.15.

4.1 Exceptions

exception SentinelAPIError(*msg="*, *response=None*)

> Bases: `Exception`

> Invalid response from DataHub. Base class for more specific exceptions.

> **msg**

> > The error message.

> > > **Type** str

> **response**

> > The response from the server as a *requests.Response* object.

> > > **Type** requests.Response

exception LTAError(*msg="*, *response=None*)

> Bases: `sentinelsat.exceptions.SentinelAPIError`

> Error raised when retrieving a product from the Long Term Archive

exception LTATriggered(*uuid*)

> Bases: `sentinelsat.exceptions.SentinelAPIError`

> Download failed due to product being in the Long Term Archive and retrieval from LTA was triggered

exception ServerError(*msg="*, *response=None*)

> Bases: `sentinelsat.exceptions.SentinelAPIError`

> Error raised when the server responded in an unexpected manner, typically due to undergoing maintenance

exception UnauthorizedError(*msg="*, *response=None*)

> Bases: `sentinelsat.exceptions.SentinelAPIError`

> Error raised when attempting to retrieve a product with incorrect credentials

exception QuerySyntaxError(*msg*, *response*)

> Bases: `sentinelsat.exceptions.SentinelAPIError`, `SyntaxError`

> Error raised when the query string could not be parsed on the server side

exception QueryLengthError(*msg=''*, *response=None*)
    Bases: *sentinelsat.exceptions.SentinelAPIError*

    Error raised when the query string length was excessively long

exception InvalidKeyError(*msg*, *response*)
    Bases: *sentinelsat.exceptions.SentinelAPIError*, KeyError

    Error raised when product with given key was not found on the server

exception InvalidChecksumError
    Bases: Exception

    MD5 checksum of a local file does not match the one from the server.

Common Issues

I keep getting *HTTP 401 Unauthorized* messages*

This means that the given username/password combination is incorrect. Note that if you created your account recently it could take a while (a few days?) until you can use that account on *apihub* and therefore `sentinelsat` too. You can go here to test access on the *apihub* endpoint.

The query fails with *HTTP 500 connection timed out*

SciHub servers are known to have outages due to high demand, try again later.

Query fails with 'Longitude/Latitude is out of bounds, check your JSON format or data.'

The server expects coordinates in the WGS84 format, check if your data complies with it.

A query to Eumetsat CODA fails with *HTTP 401 - Full authentication is required* even with the correct password.

CODA does not allow the following special characters in passwords:
~ ! @ # $ % ^ & * _ - + = ` | \ ( ) { } [ ] : ; " ' < > , . ? /
See also issue #315.

I get the warning 'The query string is too long and will likely cause a bad DHuS response'.

The query sent to the DHuS server is too complex and will likely fail. You can counter this by decreasing the query length by removing parameters or simplifying your polygon, i.e. remove vertices or decrease coordinate precision after the decimal point.

Anything else?

Make sure to check the existing issues on GitHub too.

Change Log

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

6.1 [1.2.1] – 2023-03-10

6.1.1 Fixed

- Fixed nodefilter not working with Sentinel-3 products. (#573 @valpamp)

6.1.2 Development Changes

- Added automated testing for Python 3.11. (#575)

6.2 [1.1.1] – 2021-11-23

6.2.1 CLI changes

- Show help when no CLI arguments are provided.

6.2.2 Changed

- Tests are no longer included in the Python source package to significantly reduce its size.

6.2.3 Fixed

- Fixed handling of missing *Online* OData attribute in older DHuS versions, such as https://coda.eumetsat.int. (#535 @valgur)

- Fix the `filter_node` function generated by `make_path_filter` so that node paths are no longer converted to lowercase before pattern matching. (#541 @avalentino)

- In `download_all()`, the initial check on files already existing now ignores server errors when determining the file name. Any existing files are still skipped when initiating the download. (#543 @joooeey)

6.3 [1.1.0] – 2021-08-19

`download_all()` has been completely rewritten to correctly handle server-side concurrency limits and bugs.

6.3.1 CLI changes

- New CLI option `--fmt` to specify the format of the listed products. (#526 @avalentino)

- More detailed progress info during downloads. Added progressbars for overall download and LTA retrieval progress. Product names are now included in individual download progress bars. (#532 @valgur)

- Progressbars no longer conflict with logging output. (#531 @valgur)

- Added `--fail-fast` option for downloading. (#532)

6.3.2 Added

- Added `concurrent_dl_limit` and `concurrent_lta_trigger_limit` parameters to `SentinelAPI` which are used to ensure that the server-side limit for concurrent downloads (4 for sci-hub.copernicus.eu) and the maximum number of concurrent LTA retrievals (set to 10 by default) are respected everywhere. (#532)

- Added an optional `lta_timeout` parameter to `SentinelAPI`, which sets the maximum time to wait for products to be retrieved from the LTA. Defaults to unlimited. (#532)

- Added `dl_retry_delay` parameter to `SentinelAPI`, which defaults to 10 seconds and limits the rate of download retries to give the server some time to recover. (#532)

- Added `fail_fast` option to `download_all()`. (#532)

- Added support for SHA3-256 checksums used by some newer Sentinel-2 products. (#523 @valgur)

- For more advanced use-cases, `Downloader.download_all()`, which returns a detailed `DownloadStatus` enum for each product, can now be used. (#532)

- GeoJSON geometries consisting of multiple features are now supported. (#530 @mackland)

6.3.3 Changed

- `download_all()` now waits for all LTA retrievals to finish before exiting. (#532)

- Default `n_concurrent_dl` in `download_all()` increased from 2 to 4. (#532)

- Default LTA retry period was reduced from 600 to 60 seconds. (#532)

- `SentinelProductsAPI` has been merged into `SentinelAPI` and will be removed in a future release. (#532)

- Suppressed the unnecessary server stacktrace printing for `QueryLengthError`. (#524 @valgur)

6.3.4 Fixed

- Changed logic for `_get_filename` when Content-Disposition header not present (#494 @rbrishabh)

- `download_all()` strictly tracks and limits the number of concurrent GET requests made (which includes LTA triggering) to respect server-side limits. (#532, #508 @avalentino).

- Triggering is now much more robust against server-side bugs (such as the frequent HTTP 500: NullPonterException, Postgres bugs, etc.). (#532)

- Better handling of HTTP 403 error in `trigger_offline_retrieval()` (#491 @z4zz)

- Downloading with `download_all()` can now be cleanly interrupted without the downloads remaining active in the background. (#481, #532 @valgur)

6.3.5 Development Changes

- All downloading-related functionality has been moved into a separate `Downloader` class to keep the code more organized. The existing API has been kept as-is, however. (#532)

6.4 [1.0.1] – 2021-05-25

6.4.1 Fixed

- Fixed broken triggering of offline LTA products - use GET instead of HEAD since HEAD does not appear to be triggering retrieval correctly. (#486 @valgur, related to #386)

6.5 [1.0.0] – 2021-05-19

- Dropped support for Python 2.7. Now setuptools requires Python >= 3.6.

6.5.1 CLI changes

- `--start` and `--end` CLI arguments no longer use the default values of `NOW-1DAY` and `NOW`. (#459 @valgur)

- `--footprints` now expects a path for the output file to be specified instead of using the previous default `<--path>/search_footprints.geojson`. (#462 @valgur, #407 @IpsumCapra)

- `--uuid`, `--name` and `--query` must now be repeated instead of passing a single comma-separated string. Repeated keywords get treated as an "or" expression. (#463 @valgur)

- Display DHuS server version with CLI flag `--info` (#367 @thomasyoung-audet)

- Added searching by placenames with the CLI flag `--location` (#372 @thomasyoung-audet)

- Added CLI support for `--geometry` input as a WKT or GeoJSON string (#381 @thomasyoung-audet)

- Download quicklooks directly with the CLI flag `--quicklook` (#361 @mackland)

- Got rid of the special handling of `--uuid` and `--name` CLI arguments. The product IDs are now simply passed to `api.query()` as a set. As a result they no longer ignore the date range arguments (fixes #387). (#390 @valgur)

- Added `--timeout` option with a default value of 60 seconds to avoid waiting indefinitely for a response. (#475 @valgur)

- New `--gnss` CLI flag to use the GNSS end-point for orbit products query and download (#441 @avalentino).

6.5.2 Added

- Added `setinelsat/__main__.py` (#412 @avalentino)

- Added `get_stream()` (#430 @fwfichtner)

- New `sentinelsat/products.py` module providing a "product nodes" API that allows to filter and download only selected files of the requested products (#414 @avalentino)

- Added `trigger_offline_retrieval()`. (#476 @valgur)

- Added `download_quicklooks()` and `download_all_quicklooks()` (#361 @mackland)

6.5.3 Changed

- Replaced `SentinelAPIError` exceptions with more specific types:
  - `SentinelAPIError` – the parent, catch-all exception. Only used when no other more specific exception can be applied.
  - `ServerError` – raised when the server responded in an unexpected manner, typically due to undergoing maintenance.
  - `UnauthorizedError` – raised when attempting to retrieve a product with incorrect credentials.
  - `QuerySyntaxError` – raised when the query string could not be parsed on the server side.
  - `QueryLengthError` – raised when the query string length was excessively long.
  - `InvalidKeyError` – raised when product with given key was not found on the server.
  - `InvalidChecksumError` – MD5 checksum of a local file does not match the one from the server.
  - `LTAError` – raised when triggering a retrieval from the Long Term Archive failed.
  - `LTATriggered` – raised in some cases when the product is offline and retrieval was triggered successfully.

  The new exceptions are still subclasses of `SentinelAPIError` for backwards compatibility. (#285 @valgur, @dwlsalmeida)

- Tidied up the (internal) download code logic (#378 @valgur)

- Added an "exception" field to the `failed_products` dict returned by in `.download_all()` for easier debugging (#378 @valgur)

- Fixed the server-side space handling change issue reported in #383 by quoting instead of backslash-escaping the strings. (#390 @valgur)

- Queries now use `(key:value OR key:value)` format instead of the previous `key:(value OR value)` format, which for some reason took tens of seconds per query. (#390 @valgur)

- Empty queries raise a `ValueError` immediately on the client side instead of letting the server generate it. (#390 @valgur)

- Added stricter checks for empty keyword values in queries, which would cause server-side errors. (#390 @valgur)

- Gracefully handle cancelled futures. (#448 and #479 @avalentino)

- Use the HTTP status instead of OData metadata to determine the online status of a product when downloading. This is a workaround for the rare server-side bug of the OData info for the online status being incorrect (#467). (#469 @valgur)

- `download()` now raises `LTATriggered` or `LTAError` if the requested product is offline. (#476 @valgur) This is a workaround for the rare server-side bug of the OData info for the online status being incorrect (#467). (#469 @valgur)

- Set the default query timeout to 60 seconds to avoid waiting indefinitely for a response. (#475 @valgur)

- `download_all` now returns a named tuple for clarity. (#477 @valgur)

- Search queries now use GET instead of POST requests because the Sentinel-5P Pre-Operations Data Hub does not support POST (#380). As a consequence, the search query contents are now limited to latin1 encoding instead of UTF-8. (#473 @valgur)

- Use `tqdm.auto` for nicer progress reporting in notebooks. (#480 @valgur)

6.5.4 Deprecated

- Removed `query_raw()` that was deprecated in v0.12.

6.5.5 Fixed

- Fix location information for Nominatim bounding box queries (#384)

- Get file name extension more reliably from either the HTTP header or an OData attribute. (#378, #472 @valgur)

- Updated the API Hub URL to *https://apihub.copernicus.eu/apihub/*.

- Server-side error info has become much more detailed and the client code has been updated to correctly handle that.

- `check_existing()` now determines the filename correctly for Sentinel-5 products. (@valgur #452)

- Fix accidental downloading of the whole product in memory when the product is actually available despite being marked as offline in its metadata. (#386, #421, #454 @lucadelu)

- Fixed timeout not being used in some queries. (#454 @valgur)

6.5.6 Development Changes

- Fixed failing Read The Docs builds (#370)

- Replaced Travis CI with Github Actions. Added testing on MacOS and Windows. (#438 @avalentino)

- Made tests more robust by not relying on specific products being returned where possible.

6.6 [0.14] – 2020-06-12

6.6.1 Added

- trigger retrieval of offline products from LTA, while downloading online products (#297 @gbaier)
- allow input of multiple values per query parameter as logical OR (#321 @OlgaCh)
- document CODA password limitations (#315 @nishadhka)

6.6.2 Changed

- warn users about complex queries (#290)

6.6.3 Deprecated

- discontinued support for Python <=3.4

6.6.4 Fixed

- Missing `Online` field in OData response defaults to `Online:    True` instead of raising a `KeyError` (#281 @viktorbahr)
- Missing `ContentGeometry` field in OData response defaults to `footprint:    None` instead of raising a `TypeError` (#286 #325 @lukasbindreiter)

6.6.5 Development Changes

- code formatting with *black* checked by Travis-CI (#352)
- reorganize unit tests into small groups with their own files (#287)
- reduced code duplication in unit tests by making greater use of pytest fixtures. (#287)
- force unit tests to include one of the markers 'fast', 'scihub' or 'mock_api' (#287)
- automatic return code checking of CLI tests (#287)
- Replaced direct `vcrpy` usage in unit tests with `pytest-vcr`. The `pytest` command line options changed from `--vcr disable` to `--disable-vcr` and `--vcr [use|record_new|reset]` to `--vcr-record [once|record_new|all]`. See vcrpy docs for details. (#283)

6.7 [0.13] – 2019-04-05

6.7.1 Added

- Query keywords with interval ranges now also support single-sided ranges by using `None` or `'*'` to denote no bound, for example `query(date=(None, 'NOW-1YEAR'))`. If both bounds are set to unlimited, the keyword will be removed from the query. (#210)
- Raise an exception in case of duplicate keywords present in a query. Case is ignored to match the server-side behavior. (#210)

- Support for Python 3.7
- Support for GeoJSON files with a single `Feature` without a `FeatureCollection`. (#224 @scottstanie)
- Added support for Unicode symbols in search queries. (#230)
- Raise ValueError exception if longitude is outside [-180, 180] or latitude is outside [-90, 90] (#236, #218 @Andrey-Raspopov)
- optional `timeout` attribute to avoid indefinite wait on response from the server (#256, @viktorbahr)
- Parsing the `Online`, `CreationDate` and `IngestionDate` fields of an OData response
- Trying to download an offline product from the Copernicus Open Access Hub triggers its retrieval from the long term archive. Downloading of the product is **not** scheduled.
- Added support for downloading Sentinel 5P data in the CLI via the '–sentinel 5' flag

6.7.2 Changed

- Add support in the CLI for reading credentials from *~/.netrc* and document existing functionality in the API (#90)

6.7.3 Fixed

- Spaces in query parameter values are now handled correctly be escaping them with a backslash, where appropriate. (#169, #211)
- Fixed some CLI errors not returning a non-zero exit code. (#209)
- Fixed typo for `area_relation` query parameter documentation from `'Intersection'` to `'Intersects'`. (#225 @scottstanie)
- Updated `check_query_length()` logic to match the changed server-side behavior. (#230)
- Clarify usage of GeoJSON files with CLI in docs (#229 @psal93)
- `to_geopandas()` now returns an empty GeoDataFrame for an empty product list input.

6.7.4 Development Changes

- Replaced `[test]` and `[docs]` with a single `[dev]` installation extras target. (#208)
- Adapted *.travis.yml* to build *fiona* and *pyproj* from source for Python 3.7.
- Minimum pytest version `pytest >= 3.6.3` required by `pytest-socket`.
- The existing practice of not accessing the network from unit tests, unless running with `--vcr record_new` or `--vcr reset`, is now enforced by throwing a `SocketBlockedError` in such cases. (#207)

6.8 [0.12.2] – 2018-06-20

6.8.1 Added

- made exceptions more verbose regarding optional dependencies (#176)

- CLI username, password and DHuS URL can be set with environment variables `DHUS_USER`, `DHUS_PASSWORD` and `DHUS_URL` (#184, @temal-)

- added information about known errors and DHuS issues to docs (#186, @martinber)

6.8.2 Changed

- remove hard coded product type list from cli (#190, @lenniezelk)

- Made the function signature of `count()` fully compatible with `query()`. Irrelevant parameters are simply ignored.

6.8.3 Deprecated

- environment variables `SENTINEL_USER` and `SENTINEL_PASSWORD` are superceded by `DHUS_USER` and `DHUS_PASSWORD`

6.8.4 Fixed

- Updated handling of invalid queries. An exception is raised in such cases. #168

- Fixed `order_by` parameter being ignored in queries that require multiple subqueries (that is, queries that return more than 100 products) (#200)

- Special handling of quote symbols in query strings due to a server-side error is no longer necessary and has been removed. #168

- Updated effective query length calculation in `check_query_length()` to reflect server-side changes.

- skip failing tests on optional dependency Pandas for Python 3.3 and 3.4

- Unit tests work irrespective of the directory they are run from.

6.9 [0.12.1] – 2017-10-24

6.9.1 Changed

- Made checksumming the default behavior, and removed its flag from the CLI. (@gbaier2)

6.9.2 Fixed

- set `requests` encoding to UTF8

- fixed a backwards incompatible change in the `geojson` dependency

- inconsistent documentation on the use of range parameters such as `date=`

6.10 [0.12.0] – 2017-08-10

6.10.1 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.

6.10.2 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+

6.10.3 Deprecated

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

6.10.4 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.

6.11 [0.11] – 2017-06-01

6.11.1 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.

6.12 [0.10] – 2017-05-30

6.12.1 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)

6.12.2 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.

6.12.3 Deprecated

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

6.12.4 Removed

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

6.12.5 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.

6.13 [0.9.1] – 2017-03-06

6.13.1 Added

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

6.14 [0.9] – 2017-02-26

6.14.1 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.

6.14.2 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.

6.15 [0.8.1] – 2017-02-05

6.15.1 Added

- added a changelog

6.15.2 Changed

- use logging instead of print

6.15.3 Fixed

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

6.16 [0.8] – 2017-01-27

6.16.1 Added

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

6.16.2 Changed

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

6.16.3 Removed

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

6.17 [0.7.4] – 2017-01-14

6.17.1 Added

- Travis tests for Python 3.6

6.18 [0.7.3] – 2016-12-09

6.18.1 Changed

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

6.18.2 Fixed

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

6.19 [0.7.1] – 2016-10-28

6.19.1 Added

- pagination support for query results

6.19.2 Changed

- number of query results per page set to 100

6.20 [0.6.5] – 2016-06-22

6.21 Added

- support for large queries

6.21.1 Changed

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

6.22 [0.6.4] – 2016-04-06-03

6.22.1 Changed

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

6.23 [0.6.1] – 2016-04-22

6.23.1 Added

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

6.24 [0.5.5] – 2016-01-13

6.24.1 Added

- Sentinel-2 support

6.25 [0.5.1] – 2015-12-18

6.25.1 Added

- Travis added as continuous integration service for automated testing

6.26 [0.5] – 2015-12-09

6.26.1 Added

- validate downloaded products with their MD5 checksums

6.27 [0.4.3] – 2015-11-23

6.27.1 Added

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

6.27.2 Changed

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

6.28 [0.4] – 2015-09-28

6.28.1 Added

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

6.28.2 Fixed

- CA-certificate SSL errors

6.29 [0.3] – 2015-06-10

6.29.1 Added

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

6.30 [0.1] – 2015-06-05

- first release

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

s

Index

Symbols

U