1 Installation 3
  1.1 Tests .............................................. 3
  1.2 Supported Python versions .................................. 3
  1.3 Optional dependencies .................................. 4

2 Command Line Interface 5
  2.1 Quickstart ........................................... 5
  2.2 sentinelsat .......................................... 6

3 Python API 9
  3.1 Quickstart ........................................... 9
  3.2 Authentication ....................................... 10
  3.3 Sorting & Filtering ................................... 10
  3.4 Getting Product Metadata .................................. 11
  3.5 LTA-Products ......................................... 14
  3.6 Logging ................................................ 14
  3.7 More Examples ......................................... 15
  3.8 API Reference ......................................... 15

4 Common Issues 21

5 Change Log 23
  5.1 [0.13] – 2019-04-05 ........................................ 23
  5.2 [0.12.2] – 2018-06-20 .................................. 24
  5.3 [0.12.1] – 2017-10-24 .................................. 25
  5.4 [0.12.0] – 2017-08-10 .................................. 25
  5.5 [0.11] – 2017-06-01 ..................................... 26
  5.6 [0.10] – 2017-05-30 ..................................... 27
  5.7 [0.9.1] – 2017-03-06 ..................................... 28
  5.8 [0.9] – 2017-02-26 ..................................... 28
  5.9 [0.8.1] – 2017-02-05 ..................................... 28
  5.10 [0.8] – 2017-01-27 ..................................... 29
  5.11 [0.7.4] – 2017-01-14 ..................................... 29
  5.12 [0.7.3] – 2016-12-09 ..................................... 29
  5.13 [0.7.1] – 2016-10-28 ..................................... 29
  5.14 [0.6.5] – 2016-06-22 ..................................... 30
  5.15 Added ................................................ 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Document Number</th>
<th>Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.16</td>
<td>[0.6.4]</td>
<td>2016-04-06</td>
<td>30</td>
</tr>
<tr>
<td>5.17</td>
<td>[0.6.1]</td>
<td>2016-04-22</td>
<td>30</td>
</tr>
<tr>
<td>5.18</td>
<td>[0.5.5]</td>
<td>2016-01-13</td>
<td>30</td>
</tr>
<tr>
<td>5.19</td>
<td>[0.5.1]</td>
<td>2015-12-18</td>
<td>30</td>
</tr>
<tr>
<td>5.20</td>
<td>[0.5]</td>
<td>2015-12-09</td>
<td>31</td>
</tr>
<tr>
<td>5.21</td>
<td>[0.4.3]</td>
<td>2015-11-23</td>
<td>31</td>
</tr>
<tr>
<td>5.22</td>
<td>[0.4]</td>
<td>2015-09-28</td>
<td>31</td>
</tr>
<tr>
<td>5.23</td>
<td>[0.3]</td>
<td>2015-06-10</td>
<td>31</td>
</tr>
<tr>
<td>5.24</td>
<td>[0.1]</td>
<td>2015-06-05</td>
<td>31</td>
</tr>
</tbody>
</table>

6 Indices and tables

Python Module Index
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 </path/to/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('/path/to/search_polygon.geojson'))
products = api.query(footprint,
    producttype='SLC',
    orbitdirection='ASCENDING')
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]
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 DHUS_USER=<your scihub username>
export DHUS_PASSWORD=<your scihub password>
```

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

1.2 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.
1.3 Optional dependencies

The convenience functions `to_dataframe()` and `to_geodataframe()` require pandas and/or geopandas to be present.
Sentinelsat provides a CLI \texttt{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.

\begin{verbatim}
sentinelsat -u <user> -p <password> -g <search_polygon.geojson>
\end{verbatim}

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.

\begin{verbatim}
export DHUS_USER="<user>"
export DHUS_PASSWORD="<password>"
export DHUS_URL="<api_url>"
\end{verbatim}

Alternatively, you can add them to a file \texttt{.netrc} in your user home directory.

\begin{verbatim}
machine scihub.copernicus.eu
login <user>
password <password>
\end{verbatim}

Environment variables take precedence over \texttt{.netrc}. The above command then becomes

\begin{verbatim}
sentinelsat -g <search_polygon.geojson>
\end{verbatim}

Search areas (i.e. \texttt{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 \texttt{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.

```bash
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.

```bash
sentinelsat -u <user> -p <password> -d --uuid a9048d1d-fea6-4df8-bedd-7bcb212be12e
```
or by using its filename

```bash
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%.

```bash
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.

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

2.2 sentinelsat

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

Options:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u –user</td>
<td>TEXT Username [required] (or environment variable DHUS_USER)</td>
</tr>
<tr>
<td>-p password</td>
<td>TEXT Password [required] (or environment variable DHUS_PASSWORD)</td>
</tr>
<tr>
<td>-url TEXT</td>
<td>Define another API URL. Default URL is '<a href="https://scihub.copernicus.eu/apihub/">https://scihub.copernicus.eu/apihub/</a>'.</td>
</tr>
<tr>
<td>-s –start TEXT</td>
<td>Start date of the query in the format YYYYMMDD.</td>
</tr>
<tr>
<td>-e –end TEXT</td>
<td>End date of the query in the format YYYYMMDD.</td>
</tr>
<tr>
<td>-g –geometry PATH</td>
<td>Search area geometry as GeoJSON file.</td>
</tr>
<tr>
<td>-uuid TEXT</td>
<td>Select a specific product UUID instead of a query. Multiple UUIDs can separated by commas.</td>
</tr>
<tr>
<td>-name TEXT</td>
<td>Select specific product(s) by filename. Supports wildcards.</td>
</tr>
<tr>
<td>-sentinel INT</td>
<td>Limit search to a Sentinel satellite (constellation).</td>
</tr>
<tr>
<td>-instrument TEXT</td>
<td>Limit search to a specific instrument on a Sentinel satellite.</td>
</tr>
<tr>
<td>-producttype TEXT</td>
<td>Limit search to a Sentinel product type. List of valid product types can be found under producttype here.</td>
</tr>
<tr>
<td>-cloud INT</td>
<td>Maximum cloud cover in percent. (requires --sentinel to be 2 or 3)</td>
</tr>
<tr>
<td>-order-by TEXT</td>
<td>Comma-separated list of keywords to order the result by. Prefix '-' for descending order.</td>
</tr>
<tr>
<td>-limit INT</td>
<td>Maximum number of results to return. Defaults to no limit.</td>
</tr>
<tr>
<td>-download INT</td>
<td>Download all results of the query.</td>
</tr>
<tr>
<td>-path PATH</td>
<td>Set the path where the files will be saved.</td>
</tr>
<tr>
<td>-footprints</td>
<td>Create geojson file search_footprints.geojson with footprints of the query result.</td>
</tr>
<tr>
<td>version</td>
<td>Show version number and exit.</td>
</tr>
<tr>
<td>--help</td>
<td>Show help message and exit.</td>
</tr>
</tbody>
</table>

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.
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://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('/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
(continues on next page)```
# It's 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 scihub.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.

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

```python
>>> api.query(date=('NOW-8HOURS', 'NOW'), producttype='SLC')
OrderedDict([('04548172-c64a-418f-8e83-7a4d148af1e',
              {'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>
 <gml:LinearRing>
 <gml:coordinates>0.401648,34.322010 0.876987,36.884121 -0.747357,34.664474 -1.227940,34.322010 0.401648</gml:coordinates>
 </gml:LinearRing>
 </gml:outerBoundaryIs>
 </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',
```
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.

```python
>>> api.get_product_odata('04548172-c64a-418f-8e83-7a4d148ad1f4')
{'date': datetime.datetime(2017, 4, 25, 15, 56, 12, 814000),
 'footprint': 'POLYGON((34.322010 0.401648, 0.876987, 36.540989,
 36.884121 -0.747357, 34.664474 -1.227940, 34.322010 0.401648))',
 'id': '04548172-c64a-418f-8e83-7a4d148ad1f4',
 '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-8e83-7a4d148ad1f4')/$value')
```

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

```python
>>> api.get_product_odata('04548172-c64a-418f-8e83-7a4d148ad1f4', 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">
  <gml:outerBoundaryIs>
    <gml:LinearRing>
      <gml:coordinates>0.401648,34.322010 0.876987,36.540989 36.884121 -0.747357,34.664474 -1.227940,34.322010 0.401648,34.322010 0.876987,36.540989
        36.884121 -0.747357,34.664474 -1.227940,34.322010 0.401648,34.322010 0.876987,36.540989
      </gml:coordinates>
    </gml:LinearRing>
  </gml:outerBoundaryIs>
</gml:Polygon>',
```

The SAR Antenna Subsystem (SAS) is developed and built by Astrium GmbH. It is a large foldable planar phased array antenna, which is formed by a centre panel and two antenna side wings. In deployed configuration the antenna has an overall aperture of 12.3 x 0.84 m. The antenna provides a fast electronic scanning capability in azimuth and elevation and is based on low loss and highly stable waveguide radiators built in carbon fibre technology, which are already successfully used by the TerraSAR-X radar imaging mission. The SAR Electronic Subsystem (SES) is developed and built by Astrium Ltd. It provides all radar control, IF/RF signal generation and receive data handling functions for the SAR Instrument.

The fully redundant SES is based on a channelised architecture with one transmit and two receive chains, providing an amodular approach to the generation and reception of wide-band signals and the handling of multi-polarisation modes. One key feature is the implementation of the Flexible Dynamic Block Adaptive Quantisation (FD-BAQ) data compression concept, which allows an efficient use of on-board storage resources and minimises downlink times.

3.4. Getting Product Metadata
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.

```python
product_info = api.get_product_odata(<product_id>)

if product_info['Online']:
    print('Product {} is online. Starting download.'.format(<product_id>))
    api.download(<product_id>)
else:
    print('Product {} is not online.'.format(<product_id>))
```

When trying to download an offline product with `download()` or `download_all()`, these methods will instead attempt to trigger its retrieval from the LTA.

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.basicConfig`:

```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')
```

(continues on next page)
3.7 More Examples

3.7.1 Search Sentinel 2 by tile

To search for recent Sentinel 2 imagery by MGRS tile, 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
    # products after 2017-03-31
    pp = api.query(**kw)
    products.update(pp)

api.download_all(products)
```

NB: The `tileid` parameter only works for products from April 2017 onward due to missing metadata in SciHub’s DHuS catalogue. Before that, but only from December 2016 onward (i.e. for single-tile products), you can use a `filename` pattern instead:

```python
kw['filename'] = '*_T{}_*'.format(tile)  # products after 2016-12-01
```
• **show_progressbars** *(bool)* – Whether progressbars should be shown or not, e.g. during download. Defaults to True.

• **timeout** *(float or tuple, optional)* – How long to wait for DataHub response (in seconds). Tuple (connect, read) allowed.

**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`

**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
  – yyyymm-ddThh:mm:ss.SSSZ (ISO-8601)
  – yyyymm-ddThh:mm:ssZ
  – NOW
  – NOW-<n>DAY(S) (or HOUR(S), MONTH(S), etc.)
  – NOW+<n>DAY(S)
  – yyyymm-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.

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[str, dict]`

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

Create a OpenSearch API query string.

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


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** 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[str, dict]`

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

Parameters

• id (str) – 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:

download (id, directory_path=‘’, checksum=True)
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 (str) – 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 True.

Returns  product_info – Dictionary containing the product’s info from get_product_info() as well as the path on disk.

Return type  dict
Raises `InvalidChecksumError` – If the MD5 checksum does not match the checksum on the server.

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

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.
- `checksum (bool, optional)` – If True, verify the downloaded files' integrity by checking its MD5 checksum. Throws `InvalidChecksumError` if the checksum does not match. Defaults to True.

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

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 whose retrieval from the long term archive was successfully triggered.
- `set[string]` – The list of products that failed to download.

`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 3938 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]]`

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

**Return type** polygon coordinates

### 3.8.1 Exceptions

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

**msg**
The error message.

**Type** `str`

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

**Type** `requests.Response`

```
extension sentinelsat.InvalidChecksumError
MD5 checksum of a local file does not match the one from the server.
```
CHAPTER 4

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.’  
Standard GeoJSON specification contains only WGS84 format, check if your data complies with it.

My search returns 0 results  
Maybe there are no images for the specified time period, by default *sentinelsat* will query the last 24 hours only.

Anything else?  
Make sure to check the issues on GitHub too.
All notable changes to sentinelsat will be listed here.

5.1 [0.13] – 2019-04-05

5.1.1 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
5.1.2 Changed

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

5.1.3 Fixed

- Spaces in query parameter values are now handled correctly by 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.

5.1.4 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)

5.2 [0.12.2] – 2018-06-20

5.2.1 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)

5.2.2 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.

5.2.3 Deprecated

- environment variables SENTINEL_USER and SENTINEL_PASSWORD are superceded by DHUS_USER and DHUS_PASSWORD
5.2.4 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.

5.3 [0.12.1] – 2017-10-24

5.3.1 Changed

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

5.3.2 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=

5.4 [0.12.0] – 2017-08-10

5.4.1 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.

5.4.2 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+

5.4.3 Deprecated

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

5.4.4 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.

5.5 [0.11] – 2017-06-01

5.5.1 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.

5.6 [0.10] – 2017-05-30

5.6.1 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)

5.6.2 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.

5.6.3 Deprecated

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

5.6.4 Removed

• to_dict() has been removed since it is no longer required.
• load_query() has been made private (renamed to _load_query()).
5.6.5 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.

5.7 [0.9.1] – 2017-03-06

5.7.1 Added

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

5.8 [0.9] – 2017-02-26

5.8.1 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.

5.8.2 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.

5.9 [0.8.1] – 2017-02-05

5.9.1 Added

- added a changelog

5.9.2 Changed

- use logging instead of print
5.9.3 Fixed
  • docs represent new `query` and `download_all` behaviour

5.10 [0.8] – 2017-01-27

5.10.1 Added
  • options to create new, reset or ignore vcr cassettes for testing

5.10.2 Changed
  • `query` now returns a list of search results
  • `download_all` requires the list of search results as an argument

5.10.3 Removed
  • `SentinelAPI` does not save query results as class attributes

5.11 [0.7.4] – 2017-01-14

5.11.1 Added
  • Travis tests for Python 3.6

5.12 [0.7.3] – 2016-12-09

5.12.1 Changed
  • changed `SentinelAPI max_rows` attribute to `page_size` to better reflect pagination
  • tests use `vcrpy` cassettes

5.12.2 Fixed
  • support GeoJSON polygons with optional (third) z-coordinate

5.13 [0.7.1] – 2016-10-28

5.13.1 Added
  • pagination support for query results
5.13.2 Changed

- number of query results per page set to 100

5.14 [0.6.5] – 2016-06-22

5.15 Added

- support for large queries

5.15.1 Changed

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

5.16 [0.6.4] – 2016-04-06-03

5.16.1 Changed

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

5.17 [0.6.1] – 2016-04-22

5.17.1 Added

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

5.18 [0.5.5] – 2016-01-13

5.18.1 Added

- Sentinel-2 support

5.19 [0.5.1] – 2015-12-18

5.19.1 Added

- Travis added as continuous integration service for automated testing
5.20 [0.5] – 2015-12-09

5.20.1 Added

• validate downloaded products with their MD5 checksums

5.21 [0.4.3] – 2015-11-23

5.21.1 Added

• option to select a different dhus api --url

5.21.2 Changed

• https://scihub.esa.int/apihub/ as standard url

5.22 [0.4] – 2015-09-28

5.22.1 Added

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

5.22.2 Fixed

• CA-certificate SSL errors

5.23 [0.3] – 2015-06-10

5.23.1 Added

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

5.24 [0.1] – 2015-06-05

• first release
CHAPTER 6

Indices and tables

• genindex
• modindex
• search
s
sentinelsat, 15
Index

A
api_url (sentinelsat.SentinelAPI attribute), 16

C
cHECK_FILES() (sentinelsat.SentinelAPI method), 19
cHECK_QUERY_LENGTH() (sentinelsat.SentinelAPI static method), 19
count() (sentinelsat.SentinelAPI method), 17

D
DHUS_PASSWORD, 7
DHUS_USER, 7
download() (sentinelsat.SentinelAPI method), 18
download_all() (sentinelsat.SentinelAPI method), 19

E
eNVIRONMENT_VARIABLE
    DHUS_PASSWORD, 7
    DHUS_USER, 7

F
format_query() (sentinelsat.SentinelAPI static method), 17

G
gEOJSON_TO_WKT() (in module sentinelsat), 20
gET_PRODUCT_ODATA() (sentinelsat.SentinelAPI method), 18
gET_PRODUCTS_SIZE() (sentinelsat.SentinelAPI static method), 19

I
InvalidChecksumError, 20

M
msg (sentinelsat.SentinelAPIError attribute), 20

P
page_size (sentinelsat.SentinelAPI attribute), 16

Q
query() (sentinelsat.SentinelAPI method), 16
query_raw() (sentinelsat.SentinelAPI method), 17

R
read_geojson() (in module sentinelsat), 20
response (sentinelsat.SentinelAPIError attribute), 20

S
SentinelAPI (class in sentinelsat), 15
SentinelAPIError, 20
sentinelsat (module), 15
session (sentinelsat.SentinelAPI attribute), 16

T
timeout (sentinelsat.SentinelAPI attribute), 16
to_dataframe() (sentinelsat.SentinelAPI static method), 18
to_geodataframe() (sentinelsat.SentinelAPI static method), 18
to_geojson() (sentinelsat.SentinelAPI static method), 18