py-eodms-rapi

Release 1.5.6

Kevin Ballantyne (Natural Resources Canada)

Sep 11, 2023

USER GUIDE

EODMS RAPI Client is a Python3 package used to access the REST API service provided by the Earth Observation Data Management System (EODMS) from Natural Resources Canada.

This package requires Python 3.6 or higher (it was designed using Python 3.7).

CHAPTER

ONE

INSTALLATION

The package is installed using the pip command

```
pip install py-eodms-rapi -U
```

The installation will also add the following packages:

- dateparser
- Requests
- tqdm
- geomet

The package does not require the installation of the GDAL package. However, GDAL has to be installed if you wish to use ESRI Shapefiles.

CHAPTER

TWO

EXAMPLE CODE

An example to search, order and download RCM images:

```
from eodms_rapi import EODMSRAPI

# Initialize EODMSRAPI using your EODMS account credentials
rapi = EODMSRAPI('eodms-username', 'eodms-password')

# Set a polygon of geographic centre of Canada using GeoJSON
feat = [('INTERSECTS', {"type":"Polygon", "coordinates":[[[-95.47,61.4],\
        [-97.47,61.4],[-97.47,63.4],[-95.47,63.4],[-95.47,61.4]]]})]

# Set date ranges
dates = [{"start": "20190101_000000", "end": "20210621_000000"}]

# Set search filters
filters = {'Beam Mode Type': ('LIKE', ['%50m%']),
           'Polarization': ('=', 'HH HV'),
           'Incidence Angle': ('>=', 17)}

# Set the results fields
result_fields = ['ARCHIVE_RCM.PROCESSING_FACILITY', 'RCM.ANTENNA_ORIENTATION']

# Submit search
rapi.search("RCMImageProducts", filters, feat, dates, result_fields, 2)

# Get results
rapi.set_field_convention('upper')
res = rapi.get_results('full')

# Now order the images
order_res = rapi.order(res)

# Download images to a specific destination
dest = "C:\\TEMP"
dn_res = rapi.download(order_res, dest)

# Print results
rapi.print_results(dn_res)

# Clear search results
rapi.clear_results()
```

CHAPTER

THREE

CONTENTS

3.1 Initializing the EODMSRAPI

The EODMSRAPI class is the object which contains the methods and functions used to access the EODMS REST API service.

```
from eodms_rapi import EODMSRAPI
```

Initialization of the EODMSRAPI requires entry of a password from a valid EODMS account.

Note: If you do not have an EODMS account, please visit https://www.eodms-sgdot.nrcan-rncan.gc.ca/index-en.html and click the **Register (Required to Order)** link under **Account**.

```
rapi = EODMSRAPI('eodms-username', 'eodms-password')
```

3.2 Submit an Image Search

You can perform a search on the RAPI using the `search` method of the EODMSRAPI.

However, before submitting a search, you'll have to create the items used to filter the results. The `search` function requires a **Collection** name and optional **filters**, **geometry features**, **dates**, **result fields** and **maximum results** values.

3.2.1 Collection

The Collection ID has to be specified when submitting a search.

To get a list of Collection IDs, use:

```
>>> print(rapi.get_collections(as_list=True))
| EODMSRAPI | Getting Collection information, please wait...
['NAPL', 'SGBAirPhotos', 'RCMImageProducts', 'COSMO-SkyMed1', 'Radarsat1',
↪'Radarsat1RawProducts', 'Radarsat2', 'Radarsat2RawProducts', 'RCMScienceData',
↪'TerraSarX', 'DMC', 'Gaofen-1', 'GeoEye-1', 'IKONOS', 'IRS', 'PlanetScope', 'QuickBird-
↪2', 'RapidEye', 'SPOT', 'WorldView-1', 'WorldView-2', 'WorldView-3', 'VASP']
```

3.2.2 Geometry Features

The **geometry features** are a list of tuples with each tuple containing an operator and a specified geometry (`[(<operator>, <geometry>), ...]`).

The *operator* can be **contains**, **contained by**, **crosses**, **disjoint with**, **intersects**, **overlaps**, **touches**, and **within**.

Note: The *operator* is not case sensitive. However, the *geometry* value(s) should follow the proper formatting and cases for their type (i.e. follow the proper formatting for GeoJSON, WKT, etc.).

The *geometry* can be:

Type	Info	Example
A filename	• Can be a ESRI Shapefile, KML, GML or GeoJSON • Can contain points, lines or polygons and have multiple features	`feats = [('contains', 'C:\\TEMP\\test.geojson')]`
WKT format	• Can be a point, line or polygon.	`feats = [ ('intersects', 'POINT (-75.92790414335645721 45.63414106580390239)'), ('intersects', 'POINT (-76.04462125987681986 46.23234274318053849)') ]`
GeoJSON	• The 'geometry' entry from a GeoJSON Feature. • Can be a point, line or polygon.	`('within', { "type":"Polygon", "coordinates":[ [ [-75.71484393257714,45.407703298380106], [-75.6962772564671,45.40738537380734], [-75.69343667852566,45.39264326981817], [-75.71826085966613,45.390764097853655], [-75.71484393257714,45.407703298380106] ] ] })`
Coordinates	• A list of coordinates of a polygon (ex: `[(x1, y1), (x2, y2), ...]`) • A single point (ex: `[(x1, y1)]`)	`feats = [ ('contains', [ (-75.71, 45.41), (-75.70, 45.41), (-75.69, 45.39), (-75.72, 45.39), (-75.71, 45.41) ] ) ]`

Note: The GDAL Python package is required if you wish to use shapefiles.

WKT example to get results for the easternmost and westernmost points of Canada:

```
>>> feats = [('intersects', 'POINT (-141.001944 60.306389)'), ('intersects', 'POINT (-52.
↪619444 47.523611)')]
```

3.2.3 Date Range(s)

The **date range** is either:

- A list of date range dictionaries containing a *start* and *end* key. The date values should be in format *YYYYMMDD_HHMMSS*.
- A date of a previous time interval (ex: '24 hours', '7 days'). Available intervals are 'hour', 'day', 'week', 'month' or 'year' (plural is permitted).

For example, to search for images between January 1, 2019 at midnight to September 15, 2019 at 3:35:55 PM and in the last 3 days, use:

```
>>> dates = [{"start": "20190101_000000", "end": "20190915_153555"}, "3 days"]
```

3.2.4 Query Filter(s)

The **query** variable is a dictionary containing filter titles as keys and tuples with the operator and filter value such as: `{<field>: (<operator>, <value(s)>), ...}`

Example of beam mnemonic filter: `{'Beam Mnemonic': ('like', ['16M%', '3M11'])}`

The *operator* can be one of the following: **=**, **<**, **>**, **<>**, **<=**, **>=**, **like**, **starts with**, **ends with**, or **contains**.

Note: The *operator* is not case sensitive. However, *fields* and *values* are case sensitive.

The following example will search for images with **Beam Mnemonic** that equals '3M11' or contains '16M' and with **Incidence Angle** greater than or equal to 45 degrees:

```
>>> filters = {'Beam Mnemonic': ('like', 'SC50%'), 'Incidence Angle': ('<=', '45')}
```

3.2.5 Get Available Fields

You can get a list of available query fields using the `get_available_fields` and passing the **Collection ID**.

There are 3 ways to get the available fields for a Collection using the ***name_type** argument of the `get_available_fields` function:

Value	Description	Results
empty	Gets the raw field information from the RAPI.	`print(rapi.get_available_fields('RCMImageProducts'))` … `{'search': {'Special Handling Required': {'id': 'RCM.SPECIAL_HANDLING_REQUIRED', 'datatype': 'String'}, 'Client Order Number': {'id': 'ARCHIVE_IMAGE.CLIENT_ORDER_NUMBER', 'datatype': 'String'}, ...}, 'results': {'Buyer Id': {'id': 'ARCHIVE_IMAGE.AGENCY_BUYER', 'datatype': 'Integer'}, 'Archive Visibility Start Date': {'id': 'ARCHIVE_IMAGE.ARCH_VISIBILITY_START', 'datatype': 'Date'}, ...}}`
id	Gets a list of field IDs.	`print(rapi.get_available_fields('RCMImageProducts', name_type='id'))` … `{'search': ['RCM.SPECIAL_HANDLING_REQUIRED', 'ARCHIVE_IMAGE.CLIENT_ORDER_NUMBER', ...], 'results': ['ARCHIVE_IMAGE.AGENCY_BUYER', 'ARCHIVE_IMAGE.ARCH_VISIBILITY_START', ...]}`
title	Gets a list of field names (these are used when performing a search using the	`print(rapi.get_available_fields('RCMImageProducts', name_type='title'))` …

3.2.6 Get Available Field Choices

Some fields have specific choices that the user can enter. These values are included in the `get_available_fields` *empty* results, however the function `get_field_choices` in the EODMSRAPI offers results easier to manipulate.

The `get_field_choices` function requires a **Collection ID** and an optional **field** name or ID. If no field is specified, all fields and choices for the specified Collection will be returned.

Example of choices for the Polarization field in RCM:

```
>>> rapi.get_field_choices('RCMImageProducts', 'Polarization')
['CH CV', 'HH', 'HH HV', 'HH HV VH VV', 'HH VV', 'HV', 'VH', 'VH VV', 'VV']
```

3.2.7 Result Fields

The next value to set is the **result fields**. The raw JSON results from the RAPI returns only a select few fields. For example, when searching RCM images, the RAPI only returns metadata for these Field IDs:

```
RCM.ORBIT_REL
ARCHIVE_IMAGE.PROCESSING_DATETIME
ARCHIVE_IMAGE.PRODUCT_TYPE
IDX_SENSOR.SENSOR_NAME
RCM.SBEAMFULL
RCM.POLARIZATION
RCM.SPECIAL_HANDLING_REQUIRED_R
CATALOG_IMAGE.START_DATETIME
RELATED_PRODUCTS
RCM.SPECIAL_HANDLING_INSTRUCTIONS
Metadata
RCM.DOWNLINK_SEGMENT_ID
```

If you want more fields returned, you can create a list and add Field IDs (found in the 'results' entry of the `get_available_fields` method results, in bold below) of fields you'd like included in the results JSON.

```
 >>> print(rapi.get_available_fields('RCMImageProducts'))
{'search':
    {
        [...]
    },
'results':
    {
        'Buyer Id': {'id': ' \*ARCHIVE_IMAGE.AGENCY_BUYER*\ ', 'datatype': 'Integer'},
        [...]
    }
}
```

Note: The **result fields** parameter is not necessary if you use the 'full' option when getting the results after the search; see *Get Results* for more information.

For example, the following will include the Processing Facility and Look Orientation of the images:

```
>>> result_fields = ['ARCHIVE_RCM.PROCESSING_FACILITY', 'RCM.ANTENNA_ORIENTATION']
```

3.2.8 Submit Search

Now submit the search, in this example, setting the **Collection ID** to 'RCMImageProducts' and **max results** to 100:

```
>>> rapi.search("RCMImageProducts", filters=filters, features=feats, dates=dates, result_
↪fields=result_fields, max_results=100)
| EODMSRAPI | Searching for RCMImageProducts images on RAPI
| EODMSRAPI | Querying records within 1 to 1000...
| EODMSRAPI | Number of RCMImageProducts images returned from RAPI: 9
```

3.2.9 Get Results

Before getting the results, set the field type to return:

- **camel** (default): All field names will be in lower camelcase (ex: fieldName)
- **upper**: Field names will be in upper case with underscore for spaces (ex: FIELD_NAME)
- **words**: Field names will be English words (ex: Field Name)

```
>>> rapi.set_field_convention('upper')
```

Note: Changing the field name convention does not apply when using the 'raw' parameter for the `get_results` method.

Now to get the results of your search using the `get_results` method.

There are three options for getting results:

- **raw** (default): The raw JSON data results from the RAPI. Only the basic fields and the fields you specified in the result_fields will be returned.

```
>>> print(rapi.get_results('raw'))
[
   {
            "recordId": "7822244",
            "overviewUrl": "http://was-eodms.compusult.net/wes/images/No_
↪Data_Available.png",
            "collectionId": "RCMImageProducts",
            "metadata2": [
                    {
                            "id": "RCM.ANTENNA_ORIENTATION",
                            "value": "Right",
                            "label": "Look Orientation"
                    },
                    {
                            "id": "ARCHIVE_IMAGE.PROCESSING_DATETIME",
                            "value": "2020-11-09 13:49:14 GMT",
                            "label": "Processing Date"
                    },
                    {
                            "id": "ARCHIVE_IMAGE.PRODUCT_TYPE",
                            "value": "GRD",
```

(continues on next page)

(continued from previous page)


```
                        "label": "Type"
                },
                [...]
        ],
        "rapiOrderUrl": "https://www.eodms-sgdot.nrcan-rncan.gc.ca/wes/
↪rapi/order/direct?collection=RCMImageProducts&recordId=7822244&
↪destination=fill_me_in",
        "geometry": {
                "type": "Polygon",
                "coordinates": [
                        [
                                [
                                        -111.2061013084167,
                                        62.4209316874871
                                ],
                                [
                                        -111.2710554014949,
                                        62.22606212562155
                                ],
                                [
                                        -110.6882156023417,
                                        62.18309404584561
                                ],
                                [
                                        -110.6194709629304,
                                        62.3778734605923
                                ],
                                [
                                        -111.2061013084167,
                                        62.4209316874871
                                ]
                        ]
                ]
        },
        "title": "RCM2_OK1370026_PK1375301_3_16M17_20201109_134014_HH_HV_
↪GRD",
        "orderExecuteUrl": "https://www.eodms-sgdot.nrcan-rncan.gc.ca/
↪wes/Client/?entryPoint=preview#?cseq=RCMImageProducts&record=7822244",
        "thumbnailUrl": "https://www.eodms-sgdot.nrcan-rncan.gc.ca/wes/
↪getObject?FeatureID=62f0e816-8006-4768-8f32-6ef4008e6895-7822244&
↪ObjectType=Thumbview&collectionId=RCMImageProducts",
        "metadataUrl": "https://www.eodms-sgdot.nrcan-rncan.gc.ca/wes/
↪Client/?entryPoint=resultDetails&resultId=7822244&
↪collectionId=RCMImageProducts",
        "isGeorectified": "False",
        "collectionTitle": "RCM Image Products",
        "isOrderable": "True",
        "thisRecordUrl": "https://www.eodms-sgdot.nrcan-rncan.gc.ca/wes/
↪rapi/record/RCMImageProducts/7822244",
        "metadata": [
                [
                        "Look Orientation",
```


(continues on next page)

(continued from previous page)

```
                        "Right"
                ],
                [...]
            ]
    },
    [...]
]
```

- **full**: The full metadata for each image in the results from the RAPI.

Note: When running the `get_results` function for the first time, the 'full' option will require calls to the RAPI to fetch all the metadata for each image. This can take time depending on the number of images returned from the search.

The following example is the output from the 'full' results returned from the RAPI when using the 'upper' field name convention:

```
>>> print(rapi.get_results('full'))
| EODMSRAPI | Fetching result metadata: 100%|| 29/29 [00:07<00:00, ␣
↪3.81item/s]
[
        {
                "RECORD_ID": "8572605",
                "COLLECTION_ID": "RCMImageProducts",
                "GEOMETRY": {
                        "type": "Polygon",
                        "coordinates": [
                                [
                                        [
                                                -75.87136946742638,
                                                45.53642826726489
                                        ],
                                        [
                                                -75.88537895138599,
                                                45.47880111111606
                                        ],
                                        [
                                                -75.63233378406722,
                                                45.44847937835439
                                        ],
                                        [
                                                -75.61805821213746,
                                                45.50610429149886
                                        ],
                                        [
                                                -75.87136946742638,
                                                45.53642826726489
                                        ]
                                ]
                        ]
                },
```

(continues on next page)

(continued from previous page)

```
                "TITLE": "rcm_20210407_N4549W07575",
                "COLLECTION_TITLE": "RCM Image Products",
                "IS_ORDERABLE": true,
                "THIS_RECORD_URL": "https://www.eodms-sgdot.nrcan-
↪rncan.gc.ca/wes/rapi/record/RCMImageProducts/8572605",
                "ABSOLUTE_ORBIT": "9917.0",
                "ACQUISITION_END_DATE": "2021-04-07 11:12:05 GMT",
                "ACQUISITION_START_DATE": "2021-04-07 11:12:04 GMT",
                "ARCHIVE_VISIBILITY_START_DATE": "2021-04-07␣
↪11:12:04 GMT",
                "BEAM_MNEMONIC": "FSL22",
                "BEAM_MODE_DEFINITION_ID": "422",
                [...]
                "VISIBILITY_RESTRICTION_EXPIRY_DATE": "2021-04-07␣
↪11:12:06 GMT",
                "WITHIN_ORBIT_TUBE": "true",
                "WKT_GEOMETRY": "POLYGON ((-75.8713694674264 45.
↪5364282672649 0,-75.885378951386 45.4788011111161 0,-75.
↪6323337840672 45.4484793783544 0,-75.6180582121375 45.
↪5061042914989 0,-75.8713694674264 45.5364282672649 0))"
        },
        [...]
]
```

- **geojson**: The results will be returned in GeoJSON format.

  The following example is the output from the 'geojson' results returned from the RAPI when using the 'upper' field name convention:

```
>>> print(rapi.get_results('geojson'))
| EODMSRAPI | Fetching result metadata: 100%|| 29/29 [00:07<00:00, ␣
↪3.86item/s]
{
        "type": "FeatureCollection",
        "features": [
                {
                        "type": "Feature",
                        "geometry": {
                                "type": "Polygon",
                                "coordinates": [
                                        [
                                                [
                                                        -75.
↪87136946742638,
                                                        45.
↪53642826726489
                                                ],
                                                [
                                                        -75.
↪88537895138599,
                                                        45.
↪47880111111606
                                                ],
```

(continues on next page)

(continued from previous page)

```
                                        [
                                                -75.
↪63233378406722,
                                                45.
↪44847937835439
                                        ],
                                        [
                                                -75.
↪61805821213746,
                                                45.
↪50610429149886
                                        ],
                                        [
                                                -75.
↪87136946742638,
                                                45.
↪53642826726489
                                        ]
                                ]
                        ]
                },
                "properties": {
                        "RECORD_ID": "8572605",
                        "COLLECTION_ID": "RCMImageProducts",
                        "GEOMETRY": {
                                "type": "Polygon",
                                "coordinates": [
                                        [
                                                [
                                                        -75.
↪87136946742638,
                                                        45.
↪53642826726489
                                                ],
                                                [
                                                        -75.
↪88537895138599,
                                                        45.
↪47880111111606
                                                ],
                                                [
                                                        -75.
↪63233378406722,
                                                        45.
↪44847937835439
                                                ],
                                                [
                                                        -75.
↪61805821213746,
                                                        45.
↪50610429149886
                                                ],
```

(continues on next page)

(continued from previous page)

```
                                                        [
                                                                -75.
↪87136946742638,
                                                                45.
↪53642826726489
                                                        ]
                                                ]
                                        ]
                                },
                                [...]
                                "VISIBILITY_RESTRICTION_EXPIRY_DATE
↪": "2021-04-07 11:12:06 GMT",
                                "WITHIN_ORBIT_TUBE": "true",
                                "WKT_GEOMETRY": "POLYGON ((-75.
↪8713694674264 45.5364282672649 0,-75.885378951386 45.4788011111161␣
↪0,-75.6323337840672 45.4484793783544 0,-75.6180582121375 45.
↪5061042914989 0,-75.8713694674264 45.5364282672649 0))"
                        }
                },
                [...]
        ]
}
```

```
>>> res = rapi.get_results('full')
| EODMSRAPI | Fetching result metadata: 100%|| 9/9 [00:02<00:00,  4.40item/s]
```

3.2.10 Print Results

The EODMSRAPI has a `print_results` function which will print the results in pretty print. You can pass a specific results from the RAPI to the function. If not, the 'full' results will be printed.

Note: If you haven't run `get_results` prior to `print_results`, the EODMSRAPI will first fetch the full metadata which can some time depending on the number of results.

```
>>> rapi.print_results()
[
   {
     "RECORD_ID": "13791752",
     "COLLECTION_ID": "RCMImageProducts",
 ...
```

Note: In Linux, if you get the error `UnicodeEncodeError: 'ascii' codec can't encode character...`, add `export LC_CTYPE=en_US.UTF-8` to the "~/.bashrc" file and run `source ~/.bashrc`.

3.2.11 Multiple Searches

As of v1.5.0, you can now perform more than one search and the news results will be added to the existing results.

```
>>> len(res) # This line is not needed, only to show number of results before new search
9
>>> rapi.search('RCMImageProducts', max_results=4)

| EODMSRAPI | Searching for RCMImageProducts images on RAPI
| EODMSRAPI | Querying records within 1 to 1000...
| EODMSRAPI | Number of RCMImageProducts images returned from RAPI: 4
>>> res = rapi.get_results('full')
| EODMSRAPI | Fetching result metadata: 100%|| 13/13 [00:03<00:00,  4.21item/s]
>>> len(res) # This line is not needed, only to show number of results after new search
13
```

3.2.12 Clear Results

Since each search will add to any existing results, it is important to clear the results whenever needed.

```
>>> rapi.clear_results()
```

3.2.13 Full Search Code Example

```
from eodms_rapi import EODMSRAPI

# Initialize EODMSRAPI using your EODMS account credentials
rapi = EODMSRAPI('eodms-username', 'eodms-password')

# Set features using the easternmost and westernmost points of Canada in WKT format
feats = [('intersects', 'POINT (-141.001944 60.306389)'), \
        ('intersects', 'POINT (-52.619444 47.523611)')]

# Set date ranges
dates = [{"start": "20190101_000000", "end": "20190915_153555"},
        {"start": "20201013_120000", "end": "20201113_150000"}]

# Set search filters
filters = {'Beam Mnemonic': ('like', 'SC50%'), \
           'Incidence Angle': ('<=', '45')}

# Set the results fields
result_fields = ['ARCHIVE_RCM.PROCESSING_FACILITY', 'RCM.ANTENNA_ORIENTATION']

# Submit search
rapi.search("RCMImageProducts", filters, feats, dates, result_fields, 100)

# Get results
rapi.set_field_convention('upper')
res = rapi.get_results('full')
```

(continues on next page)

(continued from previous page)

```
# Print current results
rapi.print_results(res)

# Perform another search
rapi.search('RCMImageProducts', max_results=4)
res = rapi.get_results('full')

# Clear results
rapi.clear_results()
```

3.3 Order Images

To order images using the RAPI, a POST request is submitted containing the following JSON (as an example):

```
{
    "destinations": [],
    "items": [
        {
            "collectionId": "RCMImageProducts",
            "recordId": "7822244",
            "parameters": [
                {
                    "packagingFormat": "TAR"
                },
                {
                    "NOTIFICATION_EMAIL_ADDRESS": "example@email.com"
                }
            ]
        }
    ]
}
```

So, ordering images using the EODMSRAPI requires a list of **results** (items) and optional **priority**, **parameters** and **destinations** values.

3.3.1 Results

The **results** parameter can be a list of results returned from a search session or a list of items. The **results** is required.

Each item must have: `recordId` `collectionId`

3.3.2 Priority

The **priority** can be a single string entry ("Low", "Medium", "High", or "Urgent") which will be applied to all images or a list of dictionaries containing `recordId` and `priority` value for each individual image. The **priority** is optional and the default is "Medium".

3.3.3 Parameters

The **parameters** can be a list of parameter dictionaries which will be applied to all images or a list of dictionaries containing the `recordId` and `parameters`.

Each item in the `parameters` list should be the same as how it appears in the POST request (ex: `{"packagingFormat": "TAR"}`)

You can get a list of available parameters by calling the `get_order_parameters` method of the EODMSRAPI, submitting arguments **collection** and **recordId**. The **parameters** is optional.

3.3.4 Destinations

The **destinations** is a list of destination dictionaries containing a set of items. There are 2 types of destinations, "FTP" and "Physical".

The "FTP" dictionary would look something like this:

```
{
    "type": "FTP",
    "name": "FTP Name",
    "hostname": "ftp://ftpsite.com",
    "username": "username",
    "password": "password",
    "stringValue": "ftp://username@ftpsite.com/downloads",
    "path": "downloads",
    "canEdit": "false"
}
```

The "Physical" dictionary would look like this:

```
{
    "type": "Physical",
    "name": "Destination Name",
    "customerName": "John Doe",
    "contactEmail": "example@email.com",
    "organization": "Organization Name",
    "phone": "555-555-5555",
    "addr1": "123 Fake Street",
    "addr2": "Optional",
    "addr3": "Optional",
```

(continues on next page)

(continued from previous page)

```
    "city": "Ottawa",
    "stateProv": "Ontario",
    "country": "Canada",
    "postalCode": "A1A 1A1",
    "classification": "Optional"
}
```

For more information on the destination items, visit Directly Accessing the EODMS REST API.

3.3.5 Example

Here's an example of how to submit an order to the EODMSRAPI using the previous search session:

```
params = [{"packagingFormat": "TAR"}]

order_res = rapi.order(res, priority="low", parameters=params)
```

3.4 Download Images

The *download* method of the EODMSRAPI requires:

- Either the **order results** from the *order* method or a list of **Order Item IDs**.
- A **local destination path** where the images will be downloaded.

```
dest = "C:\\TEMP"
dn_res = rapi.download(order_res, dest)
```

3.5 Examples

3.5.1 Search, Order and Download

```
from eodms_rapi import EODMSRAPI

# Initialize EODMSRAPI using your EODMS account credentials
rapi = EODMSRAPI('eodms-username', 'eodms-password')

# Set a polygon of geographic centre of Canada using GeoJSON
feat = [('INTERSECTS', {"type":"Polygon", "coordinates":[[[-95.47,61.4],\
        [-97.47,61.4],[-97.47,63.4],[-95.47,63.4],[-95.47,61.4]]]})]

# Set date ranges
dates = [{"start": "20190101_000000", "end": "20210621_000000"}]

# Set search filters
filters = {'Beam Mode Type': ('LIKE', ['%50m%']),
           'Polarization': ('=', 'HH HV'),
```

(continues on next page)

(continued from previous page)

```
            'Incidence Angle': ('>=', 17)}

# Set the results fields
result_fields = ['ARCHIVE_RCM.PROCESSING_FACILITY', 'RCM.ANTENNA_ORIENTATION']

# Submit search
rapi.search("RCMImageProducts", filters, feat, dates, result_fields, 2)

# Get results
rapi.set_field_convention('upper')
res = rapi.get_results('full')

# Now order the images
order_res = rapi.order(res)

# Download images to a specific destination
dest = "C:\\TEMP"
dn_res = rapi.download(order_res, dest)

# Print results
rapi.print_results(dn_res)
```

3.5.2 Get Available Order Parameters for an Image

```
from eodms_rapi import EODMSRAPI

# Initialize EODMSRAPI using EODMS account credentials
rapi = EODMSRAPI('username', 'password')

# Get the order parameters for RCM image with Record ID 7627902
param_res = rapi.get_order_parameters('RCMImageProducts', '7627902')

# Print the parameters
print(f"param_res: {param_res}")
```

3.5.3 Cancel an Existing Order Item

```
from eodms_rapi import EODMSRAPI

# Initialize EODMSRAPI using your EODMS account credentials
rapi = EODMSRAPI('eodms-username', 'eodms-password')

# Cancel the order item with Order ID 48188 and Order Item ID 289377
delete_res = rapi.cancel_order_item('48188', '289377')
```

3.5.4 Get a List of Available Fields for a Collection

```
from eodms_rapi import EODMSRAPI

# Initialize EODMSRAPI using your EODMS account credentials
rapi = EODMSRAPI('eodms-username', 'eodms-password')

# Get the available field information for RCMImageProducts collection
fields = rapi.get_available_fields('RCMImageProducts')
print(fields)

>>> {'search': {'Special Handling Required': {'id': 'RCM.SPECIAL_HANDLING_REQUIRED',
↪'datatype': 'String'}, ...},
    'results': {'Buyer Id': {'id': 'ARCHIVE_IMAGE.AGENCY_BUYER', 'datatype': 'Integer'},␣
↪...}
}

# Get a list of available field IDs for RCMImageProducts collection
field_ids = rapi.get_available_fields('RCMImageProducts', name_type='id')
print(field_ids)

>>> {'search': ['RCM.SPECIAL_HANDLING_REQUIRED', 'ARCHIVE_IMAGE.CLIENT_ORDER_NUMBER', ...
↪],
    'results': ['ARCHIVE_IMAGE.AGENCY_BUYER', 'ARCHIVE_IMAGE.ARCH_VISIBILITY_START', ...]
}

# Get a list of available field names used to submit searches (rapi.search())
field_titles = rapi.get_available_fields('RCMImageProducts', name_type='title')
print(field_titles)

>>> {'search': ['Special Handling Required', 'Client Order Number', 'Order Key', ...],
    'results': ['Buyer Id', 'Archive Visibility Start Date', 'Client Order Item Number',␣
↪...]
}
```

3.6 eodms_rapi package

3.6.1 eodms_rapi.eodms module

class eodms_rapi.eodms.**EODMSRAPI**(*username*, *password*, *show_timestamp=True*)

Bases: `object`

The EODMSRAPI Class containing the methods for the eodms_rapi

cancel_orderItem(*orderId*, *itemId*)

cancel_order_item(*order_id*, *item_id*)

Removes an Order Item from the EODMS using the RAPI.

Parameters

- **order_id** (*int or str*) – The Order ID of the Order Item to remove.

- **item_id** (*int or str*) – The Order Item ID of the Order Item to remove.

Returns
Returns the contents of the Delete request (always empty).

Return type
byte str

clear_results()

Clears the cumulative results.

Returns
n/a

create_destination(*dest_type*, *dest_name*, ***kwargs*)

Create a new destination using the given dest_type and dest_name.

Parameters

- **dest_type** (*str*) – The destination type, either "FTP" or "Physical".
- **dest_name** (*str*) – The destination name
- **kwargs** –

Options for FTP:
hostname: The fully qualified domain name of the target FTP server. *username*: The username used to log in to the target FTP server. *password*: The password used to log in to the target FTP server. *string_val*: A readable string representation of the whole object, typically "ftp://{user}@{host}/{path}" *path*: After logging in to the FTP server, change directory to this path (optional; default is the root directory) *can_edit*: "true" if the currently connected user is allowed to modify the properties of this Order Destination object (server to client only)

Options for Physical:
customerName: The name of the customer who will receive the order (required). *contactEmail*: An email address that can be used to contact the customer if necessary (required). *organization*: The names of any organization and organizational units necessary to identify the delivery location (optional). *phone*: A phone number that can be used to contact the customer if necessary (optional). *addrs*: A list of physical delivery addresses (1 address is required, a maximum of 3). *city*: The city or other community of the physical delivery address (required). *stateProv*: The name or code identifying the state or other administrative region required for the physical delivery address (required for most countries). *country*: The name of the country to which the product is to be delivered (required). *postalCode*: Any code that is necessary for the postal system to successfully deliver the product (required for many countries). *classification*: This is the security classification of the physical address (optional).

delete_destination(*dest_type*, *dest_name*)

Deletes a specific destination using the dest_type and dest_name.

Parameters

- **dest_type** (*str*) – The destination type, either "FTP" or "Physical".
- **dest_name** (*str*) – The destination name

download(*items*, *dest*, *wait=10.0*, *max_attempts=None*, *show_progress=True*)

Downloads a list of order items from the EODMS RAPI.

Parameters

- **items** (*list or dict*) – A list of order items returned from the RAPI.

  Example:

```
{'items': [
    {'recordId': '8023427',
     'status': 'SUBMITTED',
     'collectionId': 'RCMImageProducts',
     'itemId': '346204',
     'orderId': '50975'},
...]}
```

  or

```
[{
    'recordId': '8023427',
    'status': 'SUBMITTED',
    'collectionId': 'RCMImageProducts',
    'itemId': '346204',
    'orderId': '50975'
}, ...]
```

- **dest** (*str*) – The local download folder location.
- **wait** (*float or int*) – Sets the time to wait before checking the status of all orders.
- **max_attempts** (*int*) – The number of download attempts before stopping downloads. If None, the script will continue to check and download orders until all orders have been downloaded.
- **show_progress** – Determines whether to show progress while

downloading an image :type show_progress: bool

Returns
A list of the download (completed) items.

Return type
list

download_image(*url*, *dest_fn*, *fsize*, *show_progress=True*)

Given a list of remote and local items, download the remote data if it is not already found locally.

(Adapted from the eodms-api-client (https://pypi.org/project/eodms-api-client/) developed by Mike Brady)

Parameters

- **url** (*str*) – The download URL of the image.
- **dest_fn** (*str*) – The local destination filename for the download.
- **fsize** (*int*) – The total filesize of the image.
- **show_progress** – Determines whether to show progress while

downloading an image :type show_progress: bool

edit_destination(*dest_type*, *dest_name*, ***kwargs*)

Edits an existing destination using the given dest_type and dest_name

Parameters

- **dest_type** (*str*) – The destination type, either "FTP" or "Physical".
- **dest_name** (*str*) – The destination name
- **kwargs** –

Options for FTP:

hostname: The fully qualified domain name of the target FTP server. *username*: The username used to log in to the target FTP server. *password*: The password used to log in to the target FTP server. *string_val*: A readable string representation of the whole object, typically "ftp://{user}@{host}/{path}" *path*: After logging in to the FTP server, change directory to this path (optional; default is the root directory) *can_edit*: "true" if the currently connected user is allowed to modify the properties of this Order Destination object (server to client only)

Options for Physical:

customerName: The name of the customer who will receive the order (required). *contactEmail*: An email address that can be used to contact the customer if necessary (required). *organization*: The names of any organization and organizational units necessary to identify the delivery location (optional). *phone*: A phone number that can be used to contact the customer if necessary (optional). *addrs*: A list of physical delivery addresses (1 address is required, a maximum of 3). *city*: The city or other community of the physical delivery address (required). *stateProv*: The name or code identifying the state or other administrative region required for the physical delivery address (required for most countries). *country*: The name of the country to which the product is to be delivered (required). *postalCode*: Any code that is necessary for the postal system to successfully deliver the product (required for many countries). *classification*: This is the security classification of the physical address (optional).

get_availableFields(*collection=None*, *name_type='all'*)

get_available_fields(*collection=None*, *name_type='all'*, *ui_fields=False*)

Gets a dictionary of available fields for a collection from the RAPI.

Parameters

- **collection** (*str*) – The Collection ID.
- **name_type** (*str*) – The field name type to search for.
- **ui_fields** – Determines whether to return fields that the

EODMS UI website uses. :type ui_fields: bool

Returns

A dictionary containing the available fields for the given collection.

Return type

dict

get_collection_id(*coll*)

Gets the full collection ID using the input collection ID which can be a
substring of the collection ID.

Parameters

coll (*str*) – The collection ID to check.

Returns

The full collection ID.

Return type

str

get_collections(*as_list=False*, *opt='id'*, *redo=False*)

Gets a list of available collections for the current user.

Parameters

- **as_list** (*bool*) – Determines the type of return. If False, a dictionary will be returned. If True, only a list of collection IDs will be returned.
- **opt** (*str*) – Determines whether the list of collections should be just titles or both titles and ids.
- **redo** (*bool*) – Determines whether to get the list from the RAPI again.

Returns
Either a dictionary of collections or a list of collection IDs depending on the value of as_list.

Return type
dict

get_conv(*field*)

Converts a field name into the set naming convention (self.name_conv).

Parameters
field (*str*) – The field name.

Returns
The field name with the proper naming convention ('words',

'upper' or 'camel'). :rtype: str

get_err_msg()

Gets the error message of this class after an error occurred.

Returns
The error message

Return type
str

get_fieldChoices(*collection*, *field=None*)

get_field_choices(*collection*, *field=None*, *full=False*)

Gets the available choices for a specified field. If no choices exist, then the data type is returned.

Parameters

- **collection** (*str*) – The collection containing the field.
- **field** (*str*) – The field name or field ID.

Returns
Either a list of choices or a string containing the data type.

Return type
list or str

get_msg()

Gets the latest self.msg

Returns
The self.msg

Return type
str

get_order(*order_id*)

Gets an specified order from the EODMS RAPI.

Parameters
order_id (*str or int*) – The Order ID of the specific order.

Returns
A JSON dictionary of the specific order.

Return type
dict

get_orderItem(*itemId*)

get_orderParameters(*collection*, *recordId*)

get_order_item(*item_id*)

Submits a query to the EODMS RAPI to get a specific order item.

Parameters
item_id (*str or int*) – The Order Item ID of the image to retrieve from the RAPI.

Returns
A dictionary containing the JSON format of the results from the RAPI.

Return type
dict

get_order_parameters(*collection*, *record_id*)

Gets the list of available Order parameters for a given image record.

Parameters

- **collection** (*str*) – The Collection ID for the query.
- **record_id** (*int or str*) – The Record ID for the image.

Returns
A JSON dictionary of the order parameters.

Return type
dict

get_orders(*order_res=None*, *dtstart=None*, *dtend=None*, *max_orders=100*, *status=None*, *out_format='json'*)

Sends a query to retrieve orders from the RAPI.

Parameters

- **order_res** (*list*) – The results from an order submission. If this value is included, dtstart, dtend and max_orders will be ignored.
- **dtstart** (*datetime.datetime*) – The start date for the date range of the query.
- **dtend** (*datetime.datetime*) – The end date for the date range of the query.
- **max_orders** (*int*) – The maximum number of orders to retrieve.
- **status** (*str*) – The status of the orders to retrieve.
- **out_format** (*str*) – The format of the results.

Returns
A JSON dictionary of the query results containing the orders.

Return type
dict

`get_ordersByRecords`(*records*)

`get_orders_by_records`(*records*)

Gets a list of orders from the RAPI based on a list of records.

Parameters
records (*list*) – A list of records used to get the list of orders.

Returns
A list of results from the RAPI.

Return type
list

`get_rapiUrl`()

`get_rapi_url`()

Gets the previous URL used to query the RAPI.

return: The RAPI URL. rtype: str

`get_record`(*collection*, *record_id*, *output='full'*)

Gets an image record from the RAPI.

Parameters

- **collection** (*str*) – The Collection ID of the record.
- **record_id** (*str or int*) – The Record ID of the image.
- **output** (*str*) – The format of the results (either 'full', 'raw' or 'geojson').

`get_results`(*form='raw'*, *show_progress=True*)

Gets the self.results in a given format

Parameters

- **form** (*str*) – The type of format to return.

  Available options:

  - `raw`: Returns the JSON results straight from the RAPI.
  - **`brief`: Returns the JSON results with the 'raw' metadata but** in the field convention.
  - `full`: Returns a JSON with full metadata information.
  - **`geojson`: Returns a FeatureCollection of the results** (requires geojson package).

- **show_progress** – Determines whether to show progress while

fetching metadata :type show_progress: bool

Returns
A dictionary of the results from self.results variable.

Return type
dict

is_json(*my_json*)

Checks to see in the input item is in JSON format.

Parameters
my_json (*str*) – A string value from the requests results.

Returns
True if a valid JSON format, False if not.

Return type
boolean

log_msg(*messages*, *msg_type='info'*, *log_indent=''*, *out_indent=''*)

Logs a message to the logger.

Parameters

- **messages** (*str or list*) – Either a single message or a list of messages to log.
- **msg_type** (*str*) – The type of log ('debug', 'info', 'warning', 'error', etc.)
- **log_indent** (*str*) – The amount of indentation for the log. (ex: ' ').
- **out_indent** (*str*) – The amount of indentation for the printout. (ex: ' ').

order(*results*, *priority='Medium'*, *parameters=None*, *destinations=None*)

Sends an order to EODMS using the RAPI.

Parameters

- **results** (*list*) – A list of JSON results from the RAPI.

  The results list must contain a `collectionId` key and a `recordId` key for each image.

- **priority** (*str or list*) – Determines the priority of the order.

  If you'd like to specify a separate priority for each image, pass a list of dictionaries containing the `recordId` (matching the IDs in results) and `priority`, such as:

  ```
  [{"recordId": 7627902, "priority": "Low"}, ...]
  ```

  Priority options: "Low", "Medium", "High" or "Urgent"

- **parameters** (*list*) –

  Either a list of parameters or a list of record items.

  Use the get_order_parameters method to get a list of available parameters.

  Parameter list: `[{"|internalName|": "|value|"}, ...]`

  Example:

  ```
  [
      {"packagingFormat": "TARGZ"},
      {"NOTIFICATION_EMAIL_ADDRESS":
          "kevin.ballantyne@canada.ca"},
  ...]
  ```

Parameters for each record: ``[{"recordId": |recordId|,
"parameters": [{"|internalName|": "|value|"}, ...]}]``

Example:

```
[
    {"recordId": 7627902,
     "parameters": [{"packagingFormat": "TARGZ"},
        ...]}
]
```

- **destinations** (*list*) – A JSON representation of an array of order destinations

parse_metadata(*image_res*)

Parses the metadata results from the RAPI for better JSON.

Parameters
image_res (*dict*) – A dictionary of a single record from the RAPI.

print_message(*msg*)

print_results(*results=None*)

Pretty prints the specified results.

Parameters
results (*dict*) – A JSON of results from the RAPI.

remove_duplicate_orders(*orders*)

Removes any duplicate images from a list of orders

Parameters
orders (*list*) – A list of orders.

Returns
A unique list of orders

Return type
list

reset()

Resets specific values for the EODMSRAPI.

Returns
n/a

retrieve_destinations(*collection=None*, *record_id=None*)

Retrieves a list of order destinations for the current profile.

Parameters

- **collection** (*str*) – The Collection Id. If this value is set, then the record_id must be set as well.
- **record_id** – The Record Id for a specific image. If this value is set, then the collection must be set as well.

search(*collection*, *filters=None*, *features=None*, *dates=None*, *result_fields=None*, *max_results=None*)

Sends a search to the RAPI to search for image results.

Parameters

- **collection** (*str*) – The Collection ID for the query.

- **filters** (*dict*) – A dictionary of query filters and values in the following format:

  ```
  {"|filter title|":  ("|operator|", ["value1", "value2", ...
  ]), ...}
  ```

  Example:

  ```
  {"Beam Mnemonic": {'=': []}}
  ```

- **features** (*list*) – A list of tuples containing the operator and filenames or coordinates of features to use in the search. The features can be:
  - a filename (ESRI Shapefile, KML, GML or GeoJSON)
  - a WKT format string
  - the 'geometry' entry from a GeoJSON Feature
  - a list of coordinates (ex: `[(x1, y1), (x2, y2), ...]`)
- **dates** (*list*) – A list of date range dictionaries with keys `start` and `end`. The values of the `start` and `end` can either be a string in format `yyyymmdd_hhmmss` or a datetime.datetime object.

  Example:

  ```
  [{"start": "20201013_120000", "end": "20201013_150000"}]
  ```

- **result_fields** (*str*) – A name of a field to include in the query results.
- **max_results** (*str or int*) – The maximum number of results to return from the query.

search_url(*url*, ***kwargs*)

Submits a URL to the RAPI.

Parameters

- **url** (*str*) – A valid RAPI URL (with or without the path)
- **kwargs** (*dict*) – Options include:
 filters (dict): A dictionary of filters and values for the

  RAPI.

  features (list): A list of geometries for the query.
 dates (list): A list of date range dictionaries containing keys

  'start' and 'end'.

set_attempts(*number*)

Sets number of attempts to be made to the RAPI before the script ends.

Parameters
number (*int*) – The value for the number of attempts.

set_fieldConvention(*convention*)

set_field_convention(*convention*)

Sets the naming convention of the output fields.

Parameters
convention (*str*) – The type of naming convention for the fields.

- `words`: The label with spaces and words will be returned.
- `camel` (default): The format will be lower camel case like

'camelCase'. - `upper`: The format will be all uppercase with underscore for spaces.

set_orderTimeout(*timeout*)

set_order_timeout(*timeout*)

Sets the timeout limit for an order to the RAPI.

Parameters
timeout (*float*) – The value of the timeout in seconds.

set_queryTimeout(*timeout*)

set_query_timeout(*timeout*)

Sets the timeout limit for a query to the RAPI.

Parameters
timeout (*float*) – The value of the timeout in seconds.

set_root_url(*url*)

Sets the root URL of the RAPI.

Parameters
url (*str*) – The new URL.

class eodms_rapi.eodms.**QueryError**(*msgs*)

Bases: `object`

The QueryError class is used to store error information for a query.

get_msgs(*as_str=False*)

Gets the messages stored with the QueryError.

Parameters
as_str – Determines whether to return a string or a list of

messages. :type as_str: boolean

Returns
Either a string or a list of messages.

Return type
str or list

3.6.2 eodms_rapi.geo module

class eodms_rapi.geo.**EODMSGeo**(*eodmsrapi=None*)

Bases: `object`

The Geo class contains all the methods and functions used to perform geographic processes mainly using OGR.

add_geom(*in_src*)

Processes the source and converts it for use in the RAPI.

Parameters
in_src (*str*) – The in_src can either be:

- a filename (ESRI Shapefile, KML, GML or GeoJSON) of multiple

features - a WKT format string of a single feature - the 'geometry' entry from a GeoJSON Feature - a list of coordinates (ex: `[(x1, y1), (x2, y2), ...]`)

Returns
A string of the WKT of the feature.

Return type
str

`convert_coords`(*coord_lst*, *geom_type*)

Converts a list of points to GeoJSON format.

Parameters

- **coord_lst** (*list*) – A list of points.
- **geom_type** (*str*) – The type of geometry, either 'Point', 'LineString' or 'Polygon'.

Returns
A dictionary in the GeoJSON format.

Return type
dict

`convert_imageGeom`(*coords*, *output='array'*)

`convert_image_geom`(*coords*, *output='array'*)

Converts a list of coordinates from the RAPI to a polygon geometry, array of points or as WKT.

Parameters

- **coords** (*list*) – A list of coordinates from the RAPI results.
- **output** (*str*) – The type of return, can be 'array', 'wkt' or 'geom'.

Returns
Either a polygon geometry, WKT or array of points.

Return type
multiple types

`convert_toGeoJSON`(*results*, *output='FeatureCollection'*)

`convert_toWKT`(*in_feat*, *in_type*)

`convert_to_geojson`(*results*, *output='FeatureCollection'*)

Converts RAPI results to GeoJSON geometries.

Parameters

- **results** (*list*) – A list of results from the RAPI.
- **output** (*str*) – The output of the results (either 'FeatureCollection' or 'list' for a list of features in geojson)

Returns
A dictionary of a GeoJSON FeatureCollection.

Return type
dict

`convert_to_wkt`(*in_feat*, *in_type*)

Converts a feature into WKT format.

Parameters

- **in_feat** (*dict or list*) – The input feature, either as a GeoJSON dictionary or list of points.
- **in_type** (*str*) – The type of the input, whether it's 'json', 'list' or 'file'.

Returns
The input feature converted to WKT.

Return type
str

get_features(*in_src*)

Extracts the features from an AOI file.

Parameters
in_src (*str*) – The input filename of the AOI file. Can either be a GML, KML, GeoJSON, or Shapefile.

Returns
The AOI in WKT format.

Return type
str

process_polygon(*geom*, *t_crs*)

CHAPTER

FOUR

SUPPORT

If you have any issues or questions, please contact the EODMS Support Team at eodms-sgdot@nrcan-rncan.gc.ca.

CHAPTER

FIVE

ACKNOWLEDGEMENTS

Some code in this package is based off the EODMS API Client designed by Mike Brady.

CHAPTER

SIX

LICENSE

MIT License

PYTHON MODULE INDEX

e

INDEX

I

L

M

O

P

Q

R

S