py-eodms-rapi

Release 1.2.9

Kevin Ballantyne (Natural Resources Canada)

Nov 24, 2021

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
```

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_fieldConvention('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)
```

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_availableFields` 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_availableFields` function:

Value	Description	Results
empty	Gets the raw field information from the RAPI.	`print(rapi.get_availableFields('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_availableFields('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	`print(rapi.get_availableFields(`

3.2.6 Get Available Field Choices

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

The `get_fieldChoices` 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_fieldChoices('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_availableFields` method results, in bold below) of fields you'd like included in the results JSON.

```
 >>> print(rapi.get_availableFields('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,␣
↪resultFields=result_fields, maxResults=100)
```

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_fieldConvention('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",
                            "label": "Type"
                    },
                    [...]
```

(continues on next page)

(continued from previous page)


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


(continues on next page)

(continued from previous page)

```
            ]
    },
    [...]
]
```

- **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
                                        ]
                                ]
                        ]
                },
                "TITLE": "rcm_20210407_N4549W07575",
                "COLLECTION_TITLE": "RCM Image Products",
                "IS_ORDERABLE": true,
```

(continues on next page)

(continued from previous page)

```
                "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
                                                ],
                                                [
                                                        -75.
↪63233378406722,
```

(continues on next page)

(continued from previous page)

```
                                                        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
                                                        ],
                                                        [
                                                                -75.
↪87136946742638,
```

(continues on next page)

(continued from previous page)

```
                                                                    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')
```

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()
```

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 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)')]
```

(continues on next page)

(continued from previous page)

```
# 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_fieldConvention('upper')
res = rapi.get_results('full')

rapi.print_results(res)
```

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_orderParameters` 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_fieldConvention('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_orderParameters('RCMImageProducts', '7627902')

# Print the parameters
print("param_res: %s" % 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_orderItem('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_availableFields('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_availableFields('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_availableFields('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',␣
↪...]
}
```

CHAPTER

FOUR

SUPPORT

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