**GENERAL**

**Formatting**

Example request

*import* requests

params = {'pretty': True}

product\_id = 'SRTMGL1\_NC.003'

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/product/{0}'.*format*(product\_id),

params=params)

product\_response = response.text

*print*(product\_response)

Example response

{

"SRTMGL1\_DEM": {

"AddOffset": "",

"Available": *true*,

"DataType": "int16",

"Description": "Elevation",

"Dimensions": ["time", "lat", "lon"],

"FillValue": -32768,

"IsQA": *false*,

"Layer": "SRTMGL1\_DEM",

"OrigDataType": "int16",

"OrigValidMax": 32767,

"OrigValidMin": -32767,

"QualityLayers": "['SRTMGL1\_NUM']",

"QualityProductAndVersion": "SRTMGL1\_NUMNC.003",

"ScaleFactor": "",

"Units": "Meters",

"ValidMax": 32767,

"ValidMin": -32767,

"XSize": 3601,

"YSize": 3601

}

}

API resources that can return a large JSON response include support for an additional formatting option. The additional formatting is optional and can be enabled by specifying the pretty parameter. This behavior is disabled by default as it does increase the size of the JSON response, but it can make the response much easier to read.

| **Query Parameter** | **Description** |
| --- | --- |
| pretty | A boolean used to toggle formatting, either true or false. |

**Pagination**

Example request

*import* requests

params = {'limit': 2, 'offset': 2}

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/product',

params=params)

product\_response = response.json()

*print*(product\_response)

Example response

[{

"Product": "GPW\_UN\_Adj\_PopDensity",

"Platform": "GPW",

"Description": "UN-adjusted Population Density",

"RasterType": "Tile",

"Resolution": "1000m",

"TemporalGranularity": "Quinquennial",

"Version": "004",

"Available": *true*,

"DocLink": "http://dx.doi.org/10.7927/H4HX19NJ",

"Source": "SEDAC",

"TemporalExtentStart": "2000-01-01",

"TemporalExtentEnd": "2020-12-31",

"Deleted": *false*,

"DOI": "10.7927/H4HX19NJ",

"ProductAndVersion": "GPW\_UN\_Adj\_PopDensity.004"

}, {

"Product": "MCD12Q1",

"Platform": "Combined MODIS",

"Description": "Land Cover Type",

"RasterType": "Tile",

"Resolution": "500m",

"TemporalGranularity": "Yearly",

"Version": "006",

"Available": *true*,

"DocLink": "https://doi.org/10.5067/MODIS/MCD12Q1.006",

"Source": "LP DAAC",

"TemporalExtentStart": "2001-01-01",

"TemporalExtentEnd": "2019-12-31",

"Deleted": *false*,

"DOI": "10.5067/MODIS/MCD12Q1.006",

"ProductAndVersion": "MCD12Q1.006"

}]

API resources that can possibly return many results will provide query parameters to limit the number of results that are returned by a single request. The API utilizes a cursor-based pagination mechanism allowing you to specify a limit to the number of results returned and an offset into the desired results.

For example, you would specify limit=10&offset=0 for the first 10 results, limit=10&offset=10 for the next 10 results, etc. If the number of results returned is less than the limit number specified, you've reached the end of the list of results.

| **Query Parameter** | **Description** |
| --- | --- |
| limit | The maximum number of results to return. If the number of results returned is less than the specified limit, the end of the results has been reached. |
| offset | The number of results to skip before starting to return them. |

**Caching**

Example request

*import* requests

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/product')

*for* header *in* response.headers:

*print*('{0}: {1}'.*format*(header, response.headers[header]))

Example response

HTTP/1.1 200 OK

Date: Thu, 03 Mar 2022 22:48:28 GMT

Strict-Transport-Security: max-age=31536000; includeSubDomains; preload

Content-Type: application/json

Cache-Control: private, no-transform, must-revalidate, max-age=0

Expires: Thu, 03 Mar 2022 22:48:28 GMT

ETag: "lpdms-1645139310.0-49017-811470625"

Last-Modified: Thu, 17 Feb 2022 23:08:30 GMT

Accept-Ranges: none

X-Content-Type-Options: nosniff

X-UA-Compatible: IE=edge

X-XSS-Protection: 1; mode=block

Transfer-Encoding: chunked

Strict-Transport-Security: max-age=31536000; includeSubDomains

Example conditional request based on ETag

*import* requests

etag = '"lpdms-1588697873.0-38046-811470625"'

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/product',

headers={'If-None-Match': etag}

)

*print*(response.status\_code)

Example response code

304

Example conditional request based on Last-Modified

*import* requests

last\_modified = 'Mon, 28 Feb 2022 22:57:30 GMT'

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/product',

headers={'If-Modified-Since': last\_modified}

)

*print*(response.status\_code)

Example response code

304

API calls that return data will include ETag and Last-Modified headers. These can be used to make subsequent requests using the If-None-Match and If-Modified-Since headers respectively. If the resource matches the specified ETag or has not changed since the Last-Modified timestamp, the server will return a 304 Not Modified response.

Please note that the ETag value contains double quotation marks (e.g. "lpdms-1588697873.0-38046-811470625") and must be included in its entirety when submitting a conditional request.

**Authentication**

**Login**

*POST* **/login**

Example request

*import* requests

response = requests.post('https://appeears.earthdatacloud.nasa.gov/api/login', auth=('your-username', 'your-password'))

token\_response = response.json()

*print*(token\_response)

Example response for a successful login

{

"token\_type": "Bearer",

"token": "31ncqphv-1jpPjcTe-hgWXM2xZ1bBqQxST5pcieiHKq0cMwz8IFKOxG3FZgLQonk8hBsLV\_ruAqikYXfzWy7kw",

"expiration": "2022-02-20T20:24:27Z"

}

Example response for invalid user/password combination

{

"message": "The server could not verify that you are authorized to access the URL requested. You either supplied the wrong credentials (e.g. a bad password), or your browser doesn't understand how to supply the credentials required."

}

The API leverages the same [NASA Earthdata Login](https://urs.earthdata.nasa.gov/) as the A*pp*EEARS user interface.

Whenever a request is made to one of the secure API endpoints, [HTTP Basic Authentication](https://en.wikipedia.org/wiki/Basic_access_authentication) credentials are required. Authentication first requires a call to the login service using the NASA Earthdata Login username and password to generate a Bearer token. This Bearer token will be then used for all subsequent calls to secure endpoints. **This token will expire approximately 48 hours after being acquired.** All programs should check for an HTTP 403 Forbidden status code to successfully handle token expiration by acquiring a new token.

**Logout**

*POST* **/logout**

Example request

*import* requests

token = token\_response['token']

response = requests.post(

'https://appeears.earthdatacloud.nasa.gov/api/logout',

headers={'Authorization': 'Bearer {0}'.*format*(token)})

*print*(response.status\_code)

Example response code for a successful logout

204

Example response for a token that has already expired

{

"message": "You don't have the permission to access the requested resource. It is either read-protected or not readable by the server."

}

The API also supports disposing of an authentication prior to its expiration date/time. This is a secure endpoint which will require the Bearer token acquired by a /login call to be passed in via an Authorization header. The Authorization header must be in the form of Authorization: Bearer <your-token>.

e.g. Authorization: Bearer hgWXM2xZ1bBqQxST5pcieiHKq...

 You must replace `your-token` with your personal token.

**Tasks**

Tasks in A*pp*EEARS correspond to each request associated with your user account. Therefore, each of the calls to this service require an authentication token as described [above](https://appeears.earthdatacloud.nasa.gov/api/?python#login).

**Task object**

Parameters required for all task requests

{

"task\_type": "{task\_type}",

"task\_name": "{task\_name}",

"params":

{

"dates": [

"startDate": "{startDate}",

"endDate": "{endDate}",

"recurring": *true*,

"yearRange": [start,end]

],

"layers": [

"product": "{product\_id}",

"layer": "{layer\_name}"

]

}

}

Tasks are defined as a JSON payload with several properties that are common between area and point sample requests. One simple way to view the properties for a task JSON is to observe the output request JSON from a completed A*pp*EEARS request or make an API call to list the tasks for your account as described [below](https://appeears.earthdatacloud.nasa.gov/api/?python#list-tasks).

The following properties are required for **all** tasks.

| **Property** | **Description** |
| --- | --- |
| task\_name | The user-defined name of the task. |
| task\_type | The type of the task, either area or point. |
| params | The parameters for the request, including the required dates and layers definition, an optional additionalOptions object, and any area or point specific parameters. The dates parameter consists of a date\_object. The layers parameter consists of one or more layer\_object. If EMIT v1 data are selected, additionalOptions.orthorectify can be set to true to apply orthorectification. |

Parameters required for all area requests

{

"params":

{

"geo": "{GeoJSON}",

"output":

{

"format":

{

"type": "{file\_type}"

},

"projection": "{projection\_name}"

}

}

}

The date\_object is defined as follows:

| **Property** | **Description** |
| --- | --- |
| startDate | The start of the date range for which to extract data.  All date formats should be in MM-DD-YYYY format for non-recurring date ranges and MM-DD for recurring date ranges. |
| endDate | The end of the date range for which to extract data. |
| recurring *(optional)* | Specify true for a recurring date range. |
| yearRange *(optional)* | Specify the starting and ending years for a recurring date range (e.g. [2010, 2015]). |

The layer\_object is defined as follows:

| **Property** | **Description** |
| --- | --- |
| product | The product and version identifier for the product (e.g. MOD11A1.061). |
| layer | The name of the layer (e.g. LST\_Day\_1km). |

There are additional properties that are required for **area** requests.

| **Property** | **Description** |
| --- | --- |
| geo | A GeoJSON object defining the spatial region of interest. **The projection of any coordinates must be in a**[**geographic**](http://epsg.io/4326)**projection.** |
| output | The details about how the output files should be formatted. The file\_type must be one of geotiff or netcdf4. If EMIT v1 data are selected, the file\_type must be one of netcdf4 or envi. The projection\_name must be one of the supported projections (e.g. native, geographic). The list of supported projections can be generated via an API call described [here](https://appeears.earthdatacloud.nasa.gov/api/?python#list-projections). |

Parameters required for all point requests

{

"params":

{

"coordinates": [

"latitude": "{latitude}",

"longitude": "{longitude}",

"id": "{id}",

"category": "{category}"

]

}

}

There are additional properties that are required for **point** requests.

| **Property** | **Description** |
| --- | --- |
| coordinates | A GeoJSON object defining the spatial region of interest. **The projection of any coordinates must be in a**[**geographic**](http://epsg.io/4326)**projection.**  id *(optional)* - User-defined unique string identifier for the coordinate that can be used to identify a specific coordinate.  category *(optional)* - User-defined string category used to group one or more coordinates.  latitude - Numeric geographic latitude for the coordinate.  longitude - Numeric geographic longitude for the coordinate. |

**Task query parameters**

Parameters applicable to all task requests

task\_name={task\_name}&task\_type={task\_type}&startDate={startDate}&endDate={endDate}&recurring={recurring}&yearRange={yearRange}&layer={layer}

In addition to the JSON payload described in the [Task object](https://appeears.earthdatacloud.nasa.gov/api/?python#task-object), data passed into the URL's [query string](https://en.wikipedia.org/wiki/Query_string) can be used to specify properties for the task.

If provided, query parameter properties take precedent over any JSON payload present. For instance, if the value for the task\_name property in the task object is 'task1' and the query parameter for task\_name with a value of 'task2' is provided, the value for task\_name will be 'task2'.

Space and other white space characters **must not** be present in the query parameter values.

The parameters that are supported in the query string are listed in the tables below.

**Parameters applicable to all requests:**

| **Parameter** | **Description** |
| --- | --- |
| task\_name | The user-defined name of the task. |
| task\_type | The type of the task, either area or point. |
| startDate or start\_date | The start of the date range for which to extract data.  All date formats should be in MM-DD-YYYY format for non-recurring date ranges and MM-DD for recurring date ranges. |
| endDate or end\_date | The end of the date range for which to extract data. |
| recurring *(optional)* | Specify true for a recurring date range. |
| yearRange or year\_range *(optional)* | Specify the starting and ending years for a recurring date range (e.g. 2010,2015). |
| layer | The product and version identifier for the product and the name of the layer (e.g. MOD11A1.061,LST\_Day\_1km).  Multiple layers can be specified by using multiple layer parameters (e.g. layer=MOD11A1.061,LST\_Day\_1km, &layer=MOD11A1.061,LST\_Night\_1km). |

Parameters required for all area requests

bbox={bbox}&polygon={polygon}&file\_type={file\_type}&projection\_name={projection\_name}

**Parameters required for area requests:**

| **Parameter** | **Description** |
| --- | --- |
| bbox | A rectangular bounding box in the form of {min\_longitude},{min\_latitude},{max\_longitude},{max\_latitude} (lower left, upper right) where the longitude and latitude values are in a [geographic](http://epsg.io/4326) projection (e.g. -96.626640,43.735557,-95.624623,44.736689).  Multiple bounding boxes can be specified by using multiple bbox parameters (e.g. bbox=-96.626640,43.735557,-95.624623,44.736689 &bbox=-76.852609,38.991712,-76.851965,38.992137). |
| polygon | A polygon constructed from [geographic](http://epsg.io/4326) coordinate vertices. The value must be in the form of {coordinate\_1\_longitude},{coordinate\_1\_latitude}...{coordinate\_N\_longitude},{coordinate\_N\_latitude}, where the last coordinate specified is the same as the first coordinate, closing the polygon (e.g. -96.625589,43.735247,-96.626823,43.736084, -96.625750,43.737022,-96.623669,43.735495, -96.624956,43.734751,-96.625589,43.735247).  Multiple polygons can be specified by using multiple polygon parameters (e.g. polygon=-96.625589,43.735247,-96.626823,43.736084, -96.625750,43.737022,-96.623669,43.735495, -96.624956,43.734751,-96.625589,43.735247& polygon=-76.852455,38.991464,-76.853120,38.992023, -76.852680,38.992590,-76.851500,38.992673, -76.850760,38.991872,-76.852455,38.991464). |
| file\_type | The file format of the output files. It must be one of geotiff or netcdf4. If EMIT v1 data are selected, it must be one of netcdf4 or envi. |
| projection\_name | The name of the projection of the output files. It must be one of the supported projections (e.g. native, geographic). The list of supported projections can be generated via an API call described [here](https://appeears.earthdatacloud.nasa.gov/api/?python#list-projections). |

At least one bbox or polygon parameter is required.

Parameters required for all point requests

coordinate={coordinate}

**Parameters required for point requests:**

| **Parameter** | **Description** |
| --- | --- |
| coordinate | A coordinate in the form of {id},{category},{latitude},{longitiude} where id and category are optional (e.g. US-Ha1,DBF,42.5378,-72.1715 or 42.5378,-72.1715).  Multiple coordinates can be specified by using multiple coordinate parameters (e.g. coordinate=US-Ha1,DBF,42.5378,-72.1715&coordinate=US-Ha2,DBF,44.5378,-73.1715). |

**List tasks**

*GET* **/task**

Example request

*import* requests

token = token\_response['token']

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/task',

headers={'Authorization': 'Bearer {0}'.*format*(token)})

task\_response = response.json()

*print*(task\_response)

Example response

[{

"error": *null*,

"params": {

"dates": [{

"endDate": "08-31",

"recurring": *true*,

"startDate": "06-01",

"yearRange": [2012, 2019]

}],

"layers": [{

"layer": "ET\_500m",

"product": "MOD16A2GF.061"

}, {

"layer": "Npp\_500m",

"product": "MYD17A3HGF.061"

}]

},

"status": "done",

"created": "2022-03-01T05:02:57.528245",

"task\_id": "0974a514-13fd-4bf0-b8c9-65e6e5d6b06a",

"updated": "2022-03-01T05:04:17.034297",

"user\_id": "user@example.com",

"attempts": 1,

"estimate": {

"request\_size": 1632

},

"retry\_at": *null*,

"completed": "2022-03-01T05:04:17.033318",

"has\_swath": *false*,

"task\_name": "Point Example",

"task\_type": "point",

"api\_version": *null*,

"svc\_version": "2.71",

"web\_version": "2.71",

"size\_category": "0",

"has\_nsidc\_daac": *false*,

"expires\_on": "2022-03-31T05:04:17.034297"

}, {

"error": *null*,

"params": {

"dates": [{

"endDate": "02-01-2021",

"recurring": *false*,

"startDate": "01-01-2021",

"yearRange": [1950, 2050]

}],

"layers": [{

"layer": "LST\_Day\_1km",

"product": "MOD11A1.061"

}, {

"layer": "LST\_Night\_1km",

"product": "MOD11A1.061"

}],

"output": {

"format": {

"type": "netcdf4"

},

"projection": "native"

}

},

"status": "done",

"created": "2022-03-01T05:01:45.605326",

"summary": {

"empty": *true*,

"description": "Your request did not return any output files. This could be because no source granules match your specified parameters, or all matching granules contain only fill values (indicating no valid data)."

},

"task\_id": "2c128777-022e-4f97-96e6-35bbb4077458",

"updated": "2022-03-01T05:02:33.262517",

"user\_id": "user@example.com",

"attempts": 1,

"estimate": {

"request\_size": 62980979.5294134

},

"retry\_at": *null*,

"completed": "2022-03-01T05:02:33.261512",

"has\_swath": *false*,

"task\_name": "Area Example",

"task\_type": "area",

"api\_version": *null*,

"svc\_version": "2.71",

"web\_version": "2.71",

"size\_category": "0",

"has\_nsidc\_daac": *false*,

"expires\_on": "2022-03-31T05:02:33.262517"

}]

This API call will list all of the requests associated with your user account and supports [pagination](https://appeears.earthdatacloud.nasa.gov/api/?python#pagination). The list will automatically be sorted by date descending with the most recent requests being listed first.

You can also filter the list of tasks by specifying the following optional query parameters.

| **Filter Parameter** | **Description** |
| --- | --- |
| status | The status of the sample request (see table below). |
| task\_type | The type of sample request (e.g. point, area). |

A task can be in one of several states.

| **Status** | **Description** |
| --- | --- |
| pending | The task has been submitted and is waiting until resources are available before it begins processing. |
| processing | The task is currently processing. |
| done | The task has finished processing. |
| error | An error occurred while processing the task and all retries have been exhausted. |

If no output file is produced for a processed task, the summary object will be present. The summary object is defined as follows:

| **Property** | **Description** |
| --- | --- |
| empty | true |
| description | A textual explanation. |

**Submit task**

*POST* **/task**

Example submitting a task from a file

*import* json

*import* requests

*# load the task request from a file*

*with* *open*('sample-request.json') *as* json\_file:

task = json.load(json\_file)

*# submit the task request*

token = token\_response['token']

response = requests.post(

'https://appeears.earthdatacloud.nasa.gov/api/task',

json=task,

headers={'Authorization': 'Bearer {0}'.*format*(token)})

task\_response = response.json()

*print*(task\_response)

Example submitting a task from JSON

*import* requests

*# create the task request*

task = {

'task\_type': 'point',

'task\_name': 'my-task',

'params': {

'dates': [

{

'startDate': '01-01-2010',

'endDate': '01-31-2010'

}],

'layers': [

{

'product': 'MOD11A1.061',

'layer': 'LST\_Day\_1km'

}],

'coordinates': [

{

'latitude': 42,

'longitude': -72

}]

}

}

*# submit the task request*

token = token\_response['token']

response = requests.post(

'https://appeears.earthdatacloud.nasa.gov/api/task',

json=task,

headers={'Authorization': 'Bearer {0}'.*format*(token)})

task\_response = response.json()

*print*(task\_response)

Example submitting a task using a query string

*import* requests

*# create the task request*

task = {

'task\_type': 'point',

'task\_name': 'my-task',

'startDate': '01-01-2010',

'endDate': '01-31-2010',

'layer': 'MOD11A1.061,LST\_Day\_1km',

'coordinate': '42,-72'

}

*# submit the task request*

token = token\_response['token']

response = requests.post(

'https://appeears.earthdatacloud.nasa.gov/api/task',

params=task,

headers={'Authorization': 'Bearer {0}'.*format*(token)})

task\_response = response.json()

*print*(task\_response)

Example response for a successful submit

HTTP/1.1 202

Date: Mon, 07 Mar 2022 17:04:21 GMT

Content-Type: application/json

Content-Length: 72

Location: /api/status/ff1ee49c-daae-4576-a866-7e0c040c8c03

{

"task\_id": "7b390348-0740-45a2-af36-00bf8304cb67",

"status": "pending"

}

Example response for a invalid request

HTTP/1.0 400 Bad Request

Content-Type: application/json

Content-Length: 49

Date: Mon, 07 Mar 2022 17:04:21 GMT

{

"message": "'task\_name' is a required property"

}

This API call provides a way to submit a new request to be processed. It can accept data via JSON, query string, or a combination of both. If both are specified, data from the query string takes precedent over JSON data (see [query parameters](https://appeears.earthdatacloud.nasa.gov/api/?python#task-query-parameters) for more information).

In order for your request JSON to be accepted, it must be defined with certain required properties as described in the [task object section](https://appeears.earthdatacloud.nasa.gov/api/?python#task-object). In order for query string data to be accepted, it must be defined as described in the [task query parameters section](https://appeears.earthdatacloud.nasa.gov/api/?python#task-query-parameters).

When a request has been successfully submitted, a task\_id will be returned in the HTTP Location header with a relative URL that defines where you can access the status of the newly submitted request. The task\_id is the unique identifier used to access anything relating to the request.

Below are example JSON files that can be used to load the task request from a file:

[Point Example JSON](https://appeears.earthdatacloud.nasa.gov/api/examples/sample_point_request.json)

[Area Example JSON](https://appeears.earthdatacloud.nasa.gov/api/examples/sample_area_request.json)

**Retrieve task**

*GET* **/task/{task\_id}**

Example request

*import* requests

token = token\_response['token']

task\_id = '2c128777-022e-4f97-96e6-35bbb4077458'

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/task/{0}'.*format*(task\_id),

headers={'Authorization': 'Bearer {0}'.*format*(token)}

)

task\_response = response.json()

*print*(task\_response)

Example response

{

"error": *null*,

"params": {

"geo": {

"type": "FeatureCollection",

"features": [{

"type": "Feature",

"geometry": {

"type": "Polygon",

"coordinates": [

[

[-122.90625, 39.697801],

[-122.90625, 42.561434],

[-120.144888, 42.561434],

[-120.144888, 39.697801],

[-122.90625, 39.697801]

]

]

},

"properties": {}

}],

"fileName": "User-Drawn-Polygon"

},

"dates": [{

"endDate": "02-01-2021",

"recurring": *false*,

"startDate": "01-01-2021",

"yearRange": [1950, 2050]

}],

"layers": [{

"layer": "LST\_Day\_1km",

"product": "MOD11A1.061"

}, {

"layer": "LST\_Night\_1km",

"product": "MOD11A1.061"

}],

"output": {

"format": {

"type": "netcdf4"

},

"projection": "native"

}

},

"status": "done",

"created": "2022-03-01T05:01:45.605326",

"task\_id": "2c128777-022e-4f97-96e6-35bbb4077458",

"updated": "2022-03-01T05:02:33.262517",

"user\_id": "user@example.com",

"attempts": 1,

"estimate": {

"request\_size": 62980979.5294134

},

"retry\_at": *null*,

"completed": "2022-03-01T05:02:33.261512",

"has\_swath": *false*,

"task\_name": "Area Example",

"task\_type": "area",

"api\_version": *null*,

"svc\_version": "2.71",

"web\_version": "2.71",

"size\_category": "0",

"has\_nsidc\_daac": *false*,

"expires\_on": "2022-03-31T05:02:33.262517"

}

This API call allows you to get the details about a specific task.

**Delete task**

*DELETE* **/task/{task\_id}**

Example request

*import* requests

token = token\_response['token']

task\_id = '2c128777-022e-4f97-96e6-35bbb4077458'

response = requests.delete(

'https://appeears.earthdatacloud.nasa.gov/api/task/{0}'.*format*(task\_id),

headers={'Authorization': 'Bearer {0}'.*format*(token)})

*print*(response.status\_code)

Example response code for a successful deletion

204

This API call provides a way to delete a request.

**Status**

The status API provides information on the status of all task requests that are currently being processed for your account. Therefore, each of the calls to this service require an authentication token as described [above](https://appeears.earthdatacloud.nasa.gov/api/?python#login).

**List statuses**

*GET* **/status**

Example request

*import* requests

token = token\_response['token']

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/status',

headers={'Authorization': 'Bearer {0}'.*format*(token)})

status\_response = response.json()

*print*(status\_response)

Example response

[{

"task\_id": "7a71e792-139d-49c5-9b04-72fc95bca6bb",

"user\_id": "user@example.com",

"progress": {

"details": [

{

"step": 1,

"desc": "Initializing",

"pct\_complete": 100

},

{

"step": 2,

"desc": "Downloading",

"pct\_complete": 100

},

{

"step": 3,

"desc": "Generating Files",

"pct\_complete": 100

},

{

"step": 4,

"desc": "Extracting Stats",

"pct\_complete": 0

},

{

"step": 5,

"desc": "Finalizing",

"pct\_complete": 0

}

],

"summary": 85

},

"status\_id": "18fc2894-fbb3-4777-b751-26255cea9d70",

"status\_type": "task",

"updated": "2017-10-11T12:14:57.975000"

},

{

"task\_id": "299ec6e2-652c-456b-8303-fb4ffaf66be0",

"user\_id": "user@example.com",

"progress": {

"details": [

{

"step": 1,

"desc": "Initializing",

"pct\_complete": 100

},

{

"step": 2,

"desc": "Downloading",

"pct\_complete": 100

},

{

"step": 3,

"desc": "Generating Files",

"pct\_complete": 75

},

{

"step": 4,

"desc": "Extracting Stats",

"pct\_complete": 0

},

{

"step": 5,

"desc": "Finalizing",

"pct\_complete": 0

}

],

"summary": 65

},

"status\_id": "d6c36957-7337-4ceb-ac38-36e58038a9fb",

"status\_type": "task",

"updated": "2017-08-28T20:56:36.903000"

}]

This API call will retrieve the status information for all tasks that are currently being processed.

The status information may contain one or more of the properties below, depending on the status of the task request.

| **Property** | **Description** |
| --- | --- |
| task\_id | The unique identifier for the task. |
| user\_id | The user id for the task. |
| status\_id | The unique identifier for the status entry. |
| progress | A number indicating the summary percent complete in addition to the details listing information about each step in the process. |
| status\_type | The type of the status entry (e.g. task). |
| updated | The timestamp for when the status was last modified. |

**Task status**

*GET* **/status/{task\_id}**

Example request

*import* requests

token = token\_response['token']

task\_id = 'db4351e9-9c90-4401-8480-1ec08fe159d7'

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/status/{0}'.*format*(task\_id),

headers={'Authorization': 'Bearer {0}'.*format*(token)})

status\_response = response.json()

*print*(status\_response)

Example response

[{

"task\_id": "db4351e9-9c90-4401-8480-1ec08fe159d7",

"updated": "2022-02-23T16:05:35.346021",

"user\_id": "user@example.com",

"progress": {

"details": [{

"desc": "Initializing",

"step": 1,

"pct\_complete": 100

}, {

"desc": "Downloading",

"step": 2,

"pct\_complete": 31

}, {

"desc": "Subsetting",

"step": 3,

"pct\_complete": 0

}, {

"desc": "Generating Files",

"step": 4,

"pct\_complete": 0

}, {

"desc": "Extracting Stats",

"step": 5,

"pct\_complete": 0

}, {

"desc": "Finalizing",

"step": 6,

"pct\_complete": 0

}],

"summary": 15,

"user\_consider\_limit": *true*

},

"has\_swath": *false*,

"status\_id": "cb9a92ac-1130-4b6c-bfac-8a76fde7adb4",

"status\_type": "task"

}]

Example response for completed task

HTTP/1.1 200

Date: Mon, 07 Mar 2022 17:33:53 GMT

Content-Type: application/json

Content-Length: 32.3

Location: /api/task/db4351e9-9c90-4401-8480-1ec08fe159d7

{

"task\_id": "db4351e9-9c90-4401-8480-1ec08fe159d7",

"status": "done",

"user\_id": "user@example.com",

"updated": "2022-03-07T17:02:40.740052",

"status\_type": "task"

}

This API call will retrieve the status for the specified task. This will include the overall state as well as a percent complete for any tasks that are currently processing.

If the task status is done, a 303 See Other status code is returned along with a Location header defining where the finished task can be found. Some tools and languages will automatically redirect to the URL defined in the Location header which would retrieve the details for the task.

**Bundle**

The bundle API provides information about completed tasks for your account. Therefore, any API call to this service requires an authentication token as described [above](https://appeears.earthdatacloud.nasa.gov/api/?python#login). For tasks that have a status of done, a bundle will be generated containing all of the files that were generated as part of the task request. The bundle is associated with the task\_id that was returned when the task was submitted.

**List files**

*GET* **/bundle/{task\_id}**

Example request

*import* requests

token = token\_response['token']

task\_id = '0974a514-13fd-4bf0-b8c9-65e6e5d6b06a'

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/bundle/{0}'.*format*(task\_id),

headers={'Authorization': 'Bearer {0}'.*format*(token)}

)

bundle\_response = response.json()

*print*(bundle\_response)

Example response

{

"files": [{

"sha256": "ef38760c1fe2b17c11a9604ab4ff5b8a91afe9b291f5fcce102dad5f3fb9bf98",

"file\_id": "2dc88736-c9e8-4c6e-86ae-b33bd698be9b",

"file\_name": "Point-Example-MOD16A2GF-061-results.csv",

"file\_size": 596703,

"file\_type": "csv"

}, {

"sha256": "9dd33261d2585502b8925199efa84b96ad33fd78ada4f70612183c610a0d5798",

"file\_id": "4d2ddf7c-598e-499d-86ff-2ebe57b3121f",

"file\_name": "Point-Example-MYD17A3HGF-061-results.csv",

"file\_size": 15426,

"file\_type": "csv"

}, {

"sha256": "228e56b409ef931780f49ce083107dfc3bf8c741adf943cbe8cff08af88172e3",

"file\_id": "a39e1a4e-b853-4d81-8862-ce05583ece1b",

"file\_name": "Point-Example-granule-list.txt",

"file\_size": 87122,

"file\_type": "txt"

}, {

"sha256": "49d4b4b36d28322bd180cd81f1b86cb61fe53b9da299fd5d0be40757225e37ac",

"file\_id": "5d20c371-e82a-4917-9ef7-706511d6d7f9",

"file\_name": "Point-Example-request.json",

"file\_size": 2383,

"file\_type": "json"

}, {

"sha256": "f6b6e7e9cd873dcc9561c3268d08e1eb8ffda4c2581082d44efb02300a01e4ef",

"file\_id": "5fc452c7-5e3a-4ef5-988d-fd02c636f3c1",

"file\_name": "Point-Example-MOD16A2GF-061-metadata.xml",

"file\_size": 22691,

"file\_type": "xml"

}, {

"sha256": "b034bfe7e853773f16c4f078039c0c9e7d0a82db4f01601fef0fc186390894b2",

"file\_id": "b4493daf-b0fe-45c7-a823-e0f4bf2d490a",

"file\_name": "Point-Example-MYD17A3HGF-061-metadata.xml",

"file\_size": 22674,

"file\_type": "xml"

}, {

"sha256": "81e22573d87d626db0cbae58eef35ed7b87407727d5535c0d1b07218b20194cc",

"file\_id": "f4edb655-12de-455d-bb5d-360c8277c012",

"file\_name": "README.md",

"file\_size": 19726,

"file\_type": "txt"

}],

"created": "2022-03-01T05:03:00.887987",

"task\_id": "0974a514-13fd-4bf0-b8c9-65e6e5d6b06a",

"updated": "2022-03-01T05:04:16.995088",

"bundle\_type": "point"

}

This API call lists all of the files contained in the bundle which are available for download.

The following properties describe the bundle.

| **Property** | **Description** |
| --- | --- |
| task\_id | The unique identifier for the task. |
| files | The list of all of the files that make up the bundle.  file\_id - The unique identifier for the file.  file\_name - The name of the file.  file\_size - The size of the file in bytes.  file\_type - The content type of the file.  sha256 - The SHA-256 checksum of the file content. |
| created | The timestamp for when the bundle was created. |
| updated | The timestamp for when the bundle was last modified. |
| bundle\_type | The type of bundle, either area or point. |

**Download file**

*GET* **/bundle/{task\_id}/{file\_id}**

Example download file locally using remote name

*import* os

*import* requests

token = token\_response['token']

*# get a stream to the bundle file*

task\_id = '0974a514-13fd-4bf0-b8c9-65e6e5d6b06a'

file\_id = '4d2ddf7c-598e-499d-86ff-2ebe57b3121f'

filename = file\_id

response = requests.get(

'https://appeears.earthdatacloud.nasa.gov/api/bundle/{0}/{1}'.*format*(task\_id,file\_id),

headers={'Authorization': 'Bearer {0}'.*format*(token)},

allow\_redirects=True,

stream=True

)

*# create a destination directory to store the file in*

dest\_dir = "your-destination-directory"

filepath = os.path.join(dest\_dir, filename)

os.makedirs(os.path.dirname(filepath), exist\_ok=True)

*# write the file to the destination directory*

*with* *open*(filepath, 'wb') *as* f:

*for* data *in* response.iter\_content(chunk\_size=8192):

f.write(data)

Example response

The response in this case will be the bytes contained in the file downloaded to a filename specified by the user.

Each file in a bundle can be downloaded. Just as the task has a task\_id to identify it, each file in the bundle will also have a unique file\_id which should be used for any operation on that specific file.

When making the request to this endpoint, the response will be a 302 Found redirect, with a Location header that is an [Amazon Web Services (AWS) Simple Cloud Storage (S3) URL](https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-presigned-url.html) to your file. This URL is valid for *48 hours*.

In order to successfully download your file, ensure the tool you are using allows redirecting to this URL.

**Product**

The product API provides details about all of the products and layers available in A*pp*EEARS and does not require authentication.

**List products**

*GET* **/product**

Example request

*import* requests

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/product')

product\_response = response.json()

*# create a dictionary indexed by the product name and version*

products = {p['ProductAndVersion']: p *for* p *in* product\_response}

*print*(products['VNP09A1.001'])

Example response

{

"Product": "VNP09A1",

"Platform": "Suomi NPP VIIRS",

"Description": "Surface Reflectance",

"RasterType": "Tile",

"Resolution": "1000m",

"TemporalGranularity": "8 day",

"Version": "001",

"Available": *true*,

"DocLink": "https://doi.org/10.5067/viirs/vnp09a1.001",

"Source": "LP DAAC",

"TemporalExtentStart": "2012-01-17",

"TemporalExtentEnd": "Present",

"Deleted": *false*,

"DOI": "10.5067/VIIRS/VNP09A1.001",

"ProductAndVersion": "VNP09A1.001"

}

This API call will list all of the products and their details and supports [pagination](https://appeears.earthdatacloud.nasa.gov/api/?python#pagination).

**List layers**

*GET* **/product/{product\_id}**

Example request

*import* requests

product\_id = 'SRTMGL1\_NUMNC.003'

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/product/{0}'.*format*(product\_id))

layer\_response = response.json()

*print*(layer\_response)

Example response

{

"SRTMGL1\_NUM": {

"AddOffset": "",

"Available": *true*,

"DataType": "int16",

"Description": "Quality Assurance",

"Dimensions": [

"time",

"lat",

"lon"

],

"FillValue": 0,

"IsQA": *true*,

"Layer": "SRTMGL1\_NUM",

"OrigDataType": "int16",

"OrigValidMax": 255,

"OrigValidMin": 0,

"QualityLayers": "",

"QualityProductAndVersion": "",

"ScaleFactor": "",

"Units": "None",

"ValidMax": 255,

"ValidMin": 0,

"XSize": 3601,

"YSize": 3601

}

}

This API call will list all of the layers available for a given product. Each product is referenced by its ProductAndVersion property which is also referred to as the product\_id.

**Spatial**

The spatial API provides some helper services used to support submitting task requests.

**List projections**

*GET* **/spatial/proj**

Example request

*import* requests

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/spatial/proj')

proj\_response = response.json()

*print*(proj\_response)

Example response

[{

"Name": "native",

"Description": "Native Projection",

"Platforms": "",

"Proj4": "",

"Datum": "",

"EPSG": "",

"Units": "",

"GridMapping": "",

"Available": *true*

}, {

"Name": "geographic",

"Description": "Geographic",

"Platforms": "['SRTM', 'ECOSTRESS', 'SSEBop ET', 'GPW', 'ASTER GDEM', 'NASADEM']",

"Proj4": "+proj=longlat +datum=WGS84 +no\_defs=True",

"Datum": "wgs84",

"EPSG": 4326.0,

"Units": "degrees",

"GridMapping": "latitude\_longitude",

"Available": *true*

}, {

"Name": "sinu\_modis",

"Description": "MODIS Sinusoidal",

"Platforms": "['Combined MODIS', 'Terra MODIS', 'Aqua MODIS', 'Suomi NPP VIIRS', 'Global WELD']",

"Proj4": "+proj=sinu +lon\_0=0 +x\_0=0 +y\_0=0 +R=6371007.181 +units=m +no\_defs=True",

"Datum": "",

"EPSG": "",

"Units": "meters",

"GridMapping": "sinusoidal",

"Available": *true*

}, {

"Name": "albers\_weld\_alaska",

"Description": "WELD Albers Equal Area Alaska",

"Platforms": "['WELD']",

"Proj4": "+proj=aea +lat\_1=55 +lat\_2=65 +lat\_0=50 +lon\_0=-154 +x\_0=0 +y\_0=0 +datum=WGS84 +units=m +no\_defs=True",

"Datum": "wgs84",

"EPSG": "",

"Units": "meters",

"GridMapping": "albers\_conical\_equal\_area",

"Available": *true*

}, {

"Name": "albers\_weld\_conus",

"Description": "WELD Albers Equal Area CONUS",

"Platforms": "['WELD']",

"Proj4": "+proj=aea +lat\_1=29.5 +lat\_2=45.5 +lat\_0=23 +lon\_0=-96 +x\_0=0 +y\_0=0 +datum=WGS84 +units=m +no\_defs=True",

"Datum": "wgs84",

"EPSG": "",

"Units": "meters",

"GridMapping": "albers\_conical\_equal\_area",

"Available": *true*

}, {

"Name": "albers\_ard\_alaska",

"Description": "Landsat ARD Albers Equal Area Alaska",

"Platforms": "['Landsat ARD']",

"Proj4": "+proj=aea +lat\_1=55 +lat\_2=65 +lat\_0=50 +lon\_0=-154 +x\_0=0 +y\_0=0 +datum=WGS84 +units=m +no\_defs=True",

"Datum": "wgs84",

"EPSG": "",

"Units": "meters",

"GridMapping": "albers\_conical\_equal\_area",

"Available": *true*

}, {

"Name": "albers\_ard\_conus",

"Description": "Landsat ARD Albers Equal Area CONUS",

"Platforms": "['Landsat ARD']",

"Proj4": "+proj=aea +lat\_1=29.5 +lat\_2=45.5 +lat\_0=23 +lon\_0=-96 +x\_0=0 +y\_0=0 +datum=WGS84 +units=m +no\_defs=True",

"Datum": "wgs84",

"EPSG": "",

"Units": "meters",

"GridMapping": "albers\_conical\_equal\_area",

"Available": *true*

}, {

"Name": "albers\_ard\_hawaii",

"Description": "Landsat ARD Albers Equal Area Hawaii",

"Platforms": "['Landsat ARD']",

"Proj4": "+proj=aea +lat\_1=8 +lat\_2=18 +lat\_0=3 +lon\_0=-157 +x\_0=0 +y\_0=0 +datum=WGS84 +units=m +no\_defs=True",

"Datum": "wgs84",

"EPSG": "",

"Units": "meters",

"GridMapping": "albers\_conical\_equal\_area",

"Available": *true*

}, {

"Name": "easegrid\_2\_global",

"Description": "EASE-Grid 2.0 Global",

"Platforms": "['SMAP']",

"Proj4": "+proj=cea +lon\_0=0 +lat\_ts=30 +x\_0=0 +y\_0=0 +datum=WGS84 +units=m +no\_defs=True",

"Datum": "wgs84",

"EPSG": 6933.0,

"Units": "meters",

"GridMapping": "lambert\_cylindrical\_equal\_area",

"Available": *true*

}, {

"Name": "easegrid\_2\_north",

"Description": "EASE-Grid 2.0 Northern Hemisphere",

"Platforms": "['SMAP']",

"Proj4": "+proj=laea +lat\_0=90 +lon\_0=0 +x\_0=0 +y\_0=0 +datum=WGS84 +units=m +no\_defs=True",

"Datum": "wgs84",

"EPSG": 6931.0,

"Units": "meters",

"GridMapping": "lambert\_cylindrical\_equal\_area",

"Available": *true*

}, {

"Name": "laea\_sphere\_19",

"Description": "Lambert Azimuthal Equal Area Sphere 19",

"Platforms": "['eMODIS Smoothed NDVI']",

"Proj4": "+proj=laea +lat\_0=45 +lon\_0=-100 +x\_0=0 +y\_0=0 +a=6370997 +b=6370997 +units=m +no\_defs=True",

"Datum": "",

"EPSG": "",

"Units": "meters",

"GridMapping": "lambert\_azimuthal\_equal\_area",

"Available": *true*

}]

This API call will retrieve the list of supported projections.

**Quality**

The quality API provides quality details about all of the products available in A*pp*EEARS.

**List quality products**

*GET* **/quality**

Example request

*import* requests

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/quality')

quality\_response = response.json()

*print*(quality\_response)

Example response

[

{

"ProductAndVersion": "ASTGTM\_NC.003",

"Layer": "ASTER\_GDEM\_DEM",

"QualityProductAndVersion": "ASTGTM\_NUMNC.003",

"QualityLayers": ["ASTER\_GDEM\_NUM"],

"Continuous": *false*,

"VisibleToWorker": *true*

},

...

]

This API call will list all of the product layers and where the associated quality product and layer information is located. This API call also supports [pagination](https://appeears.earthdatacloud.nasa.gov/api/?python#pagination).

**List quality layers**

*GET* **/quality/{product\_id}**

Example request

*import* requests

product\_id = 'MOD13Q1.061'

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/quality/{0}'.*format*(product\_id))

quality\_response = response.json()

*print*(quality\_response)

Example response

[{

"ProductAndVersion": "MOD13Q1.061",

"Layer": "\_250m\_16\_days\_EVI",

"QualityProductAndVersion": "MOD13Q1.061",

"QualityLayers": ["\_250m\_16\_days\_VI\_Quality"],

"VisibleToWorker": *true*

}, {

"ProductAndVersion": "MOD13Q1.061",

"Layer": "\_250m\_16\_days\_NDVI",

"QualityProductAndVersion": "MOD13Q1.061",

"QualityLayers": ["\_250m\_16\_days\_VI\_Quality"],

"VisibleToWorker": *true*

}, {

"ProductAndVersion": "MOD13Q1.061",

"Layer": "\_250m\_16\_days\_NIR\_reflectance",

"QualityProductAndVersion": "MOD13Q1.061",

"QualityLayers": ["\_250m\_16\_days\_pixel\_reliability"],

"VisibleToWorker": *true*

}, {

"ProductAndVersion": "MOD13Q1.061",

"Layer": "\_250m\_16\_days\_blue\_reflectance",

"QualityProductAndVersion": "MOD13Q1.061",

"QualityLayers": ["\_250m\_16\_days\_pixel\_reliability"],

"VisibleToWorker": *true*

}, {

"ProductAndVersion": "MOD13Q1.061",

"Layer": "\_250m\_16\_days\_MIR\_reflectance",

"QualityProductAndVersion": "MOD13Q1.061",

"QualityLayers": ["\_250m\_16\_days\_pixel\_reliability"],

"VisibleToWorker": *true*

}, {

"ProductAndVersion": "MOD13Q1.061",

"Layer": "\_250m\_16\_days\_red\_reflectance",

"QualityProductAndVersion": "MOD13Q1.061",

"QualityLayers": ["\_250m\_16\_days\_pixel\_reliability"],

"VisibleToWorker": *true*

}]

This API call will list all of the quality layer information for a product.

**List quality values**

*GET* **/quality/{product\_id}/{quality\_layer}**

Example request

*import* requests

product\_id = 'MOD13Q1.061'

quality\_layer = '\_250m\_16\_days\_VI\_Quality'

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/quality/{0}/{1}'.*format*(product\_id, quality\_layer))

quality\_response = response.json()

*print*(quality\_response)

Example response

[

{

"ProductAndVersion": "MOD13Q1.061",

"QualityLayer": "\_250m\_16\_days\_VI\_Quality",

"Name": "MODLAND",

"Value": 0,

"Description": "VI produced with good quality",

"Acceptable": *true*

}, {

"ProductAndVersion": "MOD13Q1.061",

"QualityLayer": "\_250m\_16\_days\_VI\_Quality",

"Name": "MODLAND",

"Value": 1,

"Description": "VI produced, but check other QA",

"Acceptable": *false*

}, …

]

This API call will list all of the values for a given quality layer.

**Decode quality values**

*GET* **/quality/{product\_id}/{quality\_layer}/{quality\_value}**

Example request

*import* requests

product\_id = 'MOD13Q1.061'

quality\_layer = '\_250m\_16\_days\_VI\_Quality'

quality\_value = 0

response = requests.get('https://appeears.earthdatacloud.nasa.gov/api/quality/{0}/{1}/{2}'.*format*(product\_id, quality\_layer, quality\_value))

quality\_response = response.json()

*print*(quality\_response)

Example response

{

"Binary Representation": "0b00000000",

"Name": {

"bits": "0b00000000",

"description": "Classified land (Has a classification label and is land according to the water mask.)"

},

...

}

This API call will decode the bits for a given quality value.

**S3 Access**

**Temporary S3 Credentials**

*POST* **/s3credentials**

Example request

*import* requests

response = requests.post('https://appeears.earthdatacloud.nasa.gov/api/s3credentials', auth=('your-username', 'your-password'))

temp\_credentials = response.json()

*print*(temp\_credentials)

Example response for a successful login

{

accessKeyId: "EXAMPLE",

secretAccessKey: "EXAMPLEKEY",

sessionToken: "LONGSTRINGOFCHARACTERS...",

expiration: "2022-07-19 00:50:09+00:00"

}

Example response for invalid user/password combination

{

"message": "The server could not verify that you are authorized to access the URL requested. You either supplied the wrong credentials (e.g. a bad password), or your browser doesn't understand how to supply the credentials required."

}

This service allows a user to obtain temporary AWS [credentials](https://docs.aws.amazon.com/general/latest/gr/aws-security-credentials.html) for accessing their bundle files from AWS [cloud S3 storage](https://aws.amazon.com/s3/). It requires same [NASA Earthdata Login](https://urs.earthdata.nasa.gov/) as the A*pp*EEARS user interface i.e Whenever a request is made to one of the secure API endpoints, [HTTP Basic Authentication](https://en.wikipedia.org/wiki/Basic_access_authentication) credentials are required.

These credentials have an expiration time of 1 hour from creation. For any application using credentials from this service, a new token should be procured after the current one expires.

A variety of tools can be leveraged when using these credentials to interact with the AWS S3 API, including the [AWS CLI](https://docs.aws.amazon.com/cli/index.html) and various AWS SDKs (e.g., the [AWS Python SDK](https://aws.amazon.com/sdk-for-python/)).

**Errors**

 The error response will have an error status code listed below and always contains a description from AppEEARS in a "message" field as shown in example response

The AppEEARS API uses the following error codes:

| **Error Code** | **Meaning** |
| --- | --- |
| 400 | Bad Request -- The request could not be understood by the server due to malformed syntax. |
| 401 | Unauthorized -- Your API key is wrong. |
| 403 | Forbidden -- The api endpoint requested is only for elevated users. |
| 404 | Not Found -- The specified endpoint / resource could not be found. |
| 405 | Method Not Allowed -- You tried to access a endpoint / resource with an invalid method. |
| 429 | Too Many Requests -- The amount of requests being sent cannot be handled. Please slow the rate of your requests. |
| 500 | Internal Server Error -- We had a problem with our server. Please try again later. |
| 503 | Service Unavailable -- We're temporarily offline for maintenance. Please try again later. |

Example Response

{

"message": "You don't have the permission to access the requested resource. It is either read-protected or not readable by the server."

}