


Using Python to Query the GDC API

Python can be a versatile tool for retrieving information from the GDC API and performing downstream processing. This page details some examples that demonstrate the basic API queries using Python. The examples in this guide will use the  requests (<http://docs.python-requests.org/en/master/>) Python library and should be compatible with Python3.

Querying Metadata

Python can be used with the GDC API to retrieve metadata that is indexed in the GDC Database. See the Search and Retrieval ([../Search_and_Retrieval/](#)) section of the API documentation for specific details about parameters and usage.

A Basic Query

This example passes some basic parameters (fields, format, size) to the `cases` endpoint and prints the results in a tab-delimited format. Note that the `fields` parameter needs to be a string comprising comma-delimited field names.

Txt

Python

Choose the Python tab to view script.

Download Script ([../scripts/Basic_Query.py](#))

A Filtered Query

In the next example, a `filters` parameter is added to the script. This parameter is passed as a Python dictionary object. The filter used in this example will only display cases that come from a kidney disease study (`primary_site: Kidney`).

Txt

Python

```
import requests
import json

fields = [
    "submitter_id",
    "case_id",
    "primary_site",
    "disease_type",
    "diagnoses.vital_status"
]

fields = ",".join(fields)

cases_endpt = "https://api.gdc.cancer.gov/cases"

filters = {
    "op": "in",
    "content":{
        "field": "primary_site",
        "value": ["Kidney"]
    }
}

# With a GET request, the filters parameter needs to be converted
# from a dictionary to JSON-formatted string

params = {
    "filters": json.dumps(filters),
    "fields": fields,
    "format": "TSV",
    "size": "100"
}

response = requests.get(cases_endpt, params = params)

print(response.content)
```

[Download Script \(../scripts/Filter_Query.py\)](#)

Complex Filters

The following example uses the `and` operator in the filter to return information about files that 1) were produced using RNA-Seq, 2) are downloadable in BAM format, and 3) originate from lung cancer patients. Note that these three filters are nested within a list in the highest level `content` key.

Txt

Python

```
import requests
import json

fields = [
    "file_name",
    "cases.submitter_id",
    "cases.samples.sample_type",
    "cases.disease_type",
    "cases.project.project_id"
]

fields = ",".join(fields)

files_endpt = "https://api.gdc.cancer.gov/files"

# This set of filters is nested under an 'and' operator.
filters = {
    "op": "and",
    "content": [
        {
            "op": "in",
            "content": {
                "field": "cases.project.primary_site",
                "value": ["Lung"]
            }
        },
        {
            "op": "in",
            "content": {
                "field": "files.experimental_strategy",
                "value": ["RNA-Seq"]
            }
        },
        {
            "op": "in",
            "content": {
                "field": "files.data_format",
                "value": ["BAM"]
            }
        }
    ]
}

# A POST is used, so the filter parameters can be passed directly as a Dict object.
params = {
```

```
"filters": filters,
"fields": fields,
"format": "TSV",
"size": "2000"
}

# The parameters are passed to 'json' rather than 'params' in this case
response = requests.post(files_endpt, headers = {"Content-Type": "application/json"}, json = params)

print(response.content.decode("utf-8"))
```

[Download Script \(../scripts/Complex_Query.py\)](#)

Downloading Files

GDC files can also be downloaded from the API and saved locally using Python scripts. See the [File Download \(../Downloading_Files/\)](#) section of the API documentation for more information.

A Simple Download Request

An open-access GDC file can be downloaded by appending the file UUID to the `data` endpoint URL.

Txt

Python

```
import requests
import json
import re

file_id = "b658d635-258a-4f6f-8377-767a43771fe4"

data_endpt = "https://api.gdc.cancer.gov/data/{}".format(file_id)

response = requests.get(data_endpt, headers = {"Content-Type": "application/json"})

# The file name can be found in the header within the Content-Disposition key.
response_head_cd = response.headers["Content-Disposition"]

file_name = re.findall("filename=(.+)", response_head_cd)[0]

with open(file_name, "wb") as output_file:
    output_file.write(response.content)
```

Download Script (../scripts/Download_Files.py)

Passing a Token to Download a Controlled-Access File

A token can be passed to the script by specifying a plain text file that contains only the GDC token. A token can be downloaded by logging into the GDC Data Portal. See the Data Security (../Data/Data_Security/Data_Security) documentation for more details.

Txt

Python

```
import requests
import json

'''
    This script will not work until $TOKEN_FILE_PATH
    is replaced with an actual path.
'''
token_file = "$TOKEN_FILE_PATH"

file_id = "11443f3c-9b8b-4e47-b5b7-529468fec098"

data_endpt = "https://api.gdc.cancer.gov/slicing/view/{}".format(file_id)

with open(token_file, "r") as token:
    token_string = str(token.read().strip())

params = {"gencode": ["BRCA1", "BRCA2"]}

response = requests.post(data_endpt,
                        data = json.dumps(params),
                        headers = {
                            "Content-Type": "application/json",
                            "X-Auth-Token": token_string
                        })

file_name = "brca_slices.bam"

with open(file_name, "wb") as output_file:
    output_file.write(response.content)
```

[Download Script \(../scripts/Download_Files_Token.py\)](#)

Post Request to Download Multiple Files

This example uses a Python list to specify a set of file UUIDs. The list in the example was populated manually but could potentially be populated programmatically from an external list or API call.

Txt

Python

```
import requests
import json
import re

data_endpt = "https://api.gdc.cancer.gov/data"

ids = [
    "b658d635-258a-4f6f-8377-767a43771fe4",
    "3968213d-b293-4b3d-8033-5b5a0ca07b6c"
]

params = {"ids": ids}

response = requests.post(data_endpt,
                        data = json.dumps(params),
                        headers={
                            "Content-Type": "application/json"
                        })

response_head_cd = response.headers["Content-Disposition"]

file_name = re.findall("filename=(.+)", response_head_cd)[0]

with open(file_name, "wb") as output_file:
    output_file.write(response.content)
```

[Download Script \(../scripts/Download_Files_Post.py\)](#)

Downloading a Set of Files Based on a Filter

Here a list of files based on a set of filters are downloaded. File UUIDs are retrieved based on the filters. These UUIDs are then passed to the `data` endpoint to download the correct files.

Txt

Python


```
import requests
import json
import re

files_endpt = "https://api.gdc.cancer.gov/files"

filters = {
    "op": "and",
    "content": [
        {
            "op": "in",
            "content": {
                "field": "cases.project.primary_site",
                "value": ["Lung"]
            }
        },
        {
            "op": "in",
            "content": {
                "field": "cases.demographic.race",
                "value": ["white"]
            }
        },
        {
            "op": "in",
            "content": {
                "field": "cases.demographic.gender",
                "value": ["female"]
            }
        },
        {
            "op": "in",
            "content": {
                "field": "files.analysis.workflow_type",
                "value": ["HTSeq - FPKM"]
            }
        }
    ]
}

# Here a GET is used, so the filter parameters should be passed as a JSON string.

params = {
    "filters": json.dumps(filters),
    "fields": "file_id",
    "format": "JSON",
```

```
"size": "1000"
}

response = requests.get(files_endpt, params = params)

file_uuid_list = []

# This step populates the download list with the file_ids from the previous query
for file_entry in json.loads(response.content.decode("utf-8"))["data"]["hits"]:
    file_uuid_list.append(file_entry["file_id"])

data_endpt = "https://api.gdc.cancer.gov/data"

params = {"ids": file_uuid_list}

response = requests.post(data_endpt, data = json.dumps(params), headers = {"Content-Type": "application/json"})

response_head_cd = response.headers["Content-Disposition"]

file_name = re.findall("filename=(.+)", response_head_cd)[0]

with open(file_name, "wb") as output_file:
    output_file.write(response.content)
```

Download Script (../scripts/Download_Files_Filter.py)

BAM Slicing

The GDC BAM Slicing (../BAM_Slicing/) feature can also be accessed through Python. Below is an example of a basic BAM slicing command.

Txt

Python

```
import requests
import json

'''
    This script will not work until $TOKEN_FILE_PATH
    is replaced with an actual path.
'''
token_file = "$TOKEN_FILE_PATH"

file_id = "11443f3c-9b8b-4e47-b5b7-529468fec098"

data_endpt = "https://api.gdc.cancer.gov/slicing/view/{}".format(file_id)

with open(token_file, "r") as token:
    token_string = str(token.read().strip())

params = {"gencode": ["BRCA1", "BRCA2"]}

response = requests.post(data_endpt,
                        data = json.dumps(params),
                        headers = {
                            "Content-Type": "application/json",
                            "X-Auth-Token": token_string
                        })

file_name = "brca_slices.bam"

with open(file_name, "wb") as output_file:
    output_file.write(response.content)
```

[Download Script \(../scripts/BAM_Slice.py\)](#)

The same region(s) across multiple BAM files can be retrieved using a for-loop within a Python script.

Txt

Python

```
import requests
import json

'''
This script will not work until $TOKEN_FILE_PATH
is replaced with an actual path.
'''
token_file = "$TOKEN_FILE_PATH"

file_ids = [
    "11443f3c-9b8b-4e47-b5b7-529468fec098",
    "1f103620-bb34-46f1-b565-94f0027e396d",
    "ca549554-a244-4209-9086-92add7bb7109"
]

for file_id in file_ids:

    data_endpt = "https://api.gdc.cancer.gov/slicing/view/{}".format(file_id)

    with open(token_file, "r") as token:
        token_string = str(token.read().strip())

    params = {
        "regions": ["chr1:1-20000", "chr10:129000-160000"]
    }

    response = requests.post(data_endpt,
                             data = json.dumps(params),
                             headers = {
                                 "Content-Type": "application/json",
                                 "X-Auth-Token": token_string
                             })

    file_name = "{}_region_slices.bam".format(file_id)

    with open(file_name, "wb") as output_file:
        output_file.write(response.content)
```

[Download Script \(../scripts/BAM_Slice_Multiple.py\)](#)

Basic Troubleshooting

The following script should produce an unformatted JSON string with information about the API status. Run this script to verify that a valid connection is being made to the GDC API.

Python

```
import requests
status_endpt = "https://api.gdc.cancer.gov/status"
response = requests.get(status_endpt)
print(response.content)
```

◀ Previous: Submission (../Submission/)

Next: System Information ▶ (../System_Information/)

Site Home (<https://portal.gdc.cancer.gov>) | Policies (<http://www.cancer.gov/global/web/policies>) | Accessibility (<http://www.cancer.gov/global/web/policies/accessibility>) | FOIA (<http://www.cancer.gov/global/web/policies/foia>)

U.S. Department of Health and Human Services (<http://www.hhs.gov>) | National Institutes of Health (<http://www.nih.gov>) | National Cancer Institute (<http://www.cancer.gov>) | USA.gov (<http://www.usa.gov>)

NIH... Turning Discovery Into Health ®

GDC Docs Version 1.0