Query Tool Application Programming Interface (API)
Table of Contents

Query Tool Application Programming Interface ........................................... 3

Objective ........................................................................................................... 3

Authentication ................................................................................................. 4

Authentication ................................................................................................. 4

Study API ......................................................................................................... 5

Get All Study Information ................................................................................. 5

Get all studies associated with a form structure .............................................. 7

Form Structure API .......................................................................................... 8

Get form structures for a study ....................................................................... 8

Data Elements API ............................................................................................ 10

Get data elements for a form structure .......................................................... 10

Data API ............................................................................................................ 12

Get data from multiple form structures without doing joins ....................... 12

Get data from multiple form structures for the given studies ..................... 14

Get data with filter and joins ........................................................................... 15
QUERY TOOL APPLICATION PROGRAMMING INTERFACE

The Query Tool Application Programming Interface (API), a RESTful API that is connected to a micro service that allows users to make HTTPS requests using GET and POST. Users will be able to query data from different endpoints using Python, JavaScript, R and other tools that use Restful APIs.

OBJECTIVE

This chapter provides information for users on how to:

- To log in to the Query Tool API
- Enter parameter information for each endpoint

For information about the Query Tool API Endpoints please access the API Documentation for the BRICS Instance:

<table>
<thead>
<tr>
<th>BRICS Instance</th>
<th>Authentication API</th>
<th>Query API</th>
</tr>
</thead>
</table>

For more information and examples on the endpoints, please refer to the Jupyter Notebook.
AUTHENTICATION

In order to use the Query Tool API, the user first must POST their credentials (i.e., Username and Password) for the Authentication endpoint.

Once, the user enters his/her credentials, a token will be provided, which will be used for all subsequent endpoints. If the token expires, the user can retrieve a new token or renew it. The section on Authentication, will provide more information entering credentials and retrieving the token using HTTPS requests with Python.

AUTHENTICATION

To log in the Query Tool API the user needs to log in and retrieve the access token that is used for subsequent endpoints.

This service will authenticate a user permission to use BRICS Query Tool API.

The following information is needed:

Endpoint URL: https://bricsnei-stage.cit.nih.gov/gateway/authentication/user/login

Parameters:

<table>
<thead>
<tr>
<th>Headers</th>
<th>Response content type: text/plain'</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Type: application/x-www-form-urlencoded</td>
</tr>
<tr>
<td>Data</td>
<td>Username</td>
</tr>
<tr>
<td></td>
<td>Password</td>
</tr>
</tbody>
</table>

Figure 1: Example of input information

```python
# login in to API
url = "https://bricsnei-stage.cit.nih.gov/gateway/authentication/user/login"
headers = {
    'accept': 'text/plain',
    'Content-Type': 'application/x-www-form-urlencoded'
}
data = {'password':UserPassword,
        'username': UserUsername}

response = requests.post(url, headers=headers, data=data)
```
Output

Output Format | Text
---|---

**Figure 2: Example of output Information**

```python
login = check
if response.status_code == 200:
    print("Login Successful")
    token = response.text
else:
    print("Error, Please check username and password. If error still occurs reach out to system administrator. THIS CODE WILL NOT PROCEED")
```

Login Successful
Here is your token: ay1hbbol10m20w125x:ay2jke10loj3c00v9w9m4d40y2ds7e213p1la0gic0137w9v18eb1234678eb0123456k2vau9wvdf

**STUDY API**

The following endpoints will return the study profile information for a study. As mentioned in the Authentication section, a token is needed to retrieve data for all subsequent endpoints.

There are two endpoints that allow users to the do following:

1. Get study profile information for all studies.
2. Get study profile information for one study using a Study Prefix.
3. Get studies that have submitted data to a form structure using the Form Structure Shortname.

Below are the endpoints for retrieving information about studies and examples.

**GET ALL STUDY INFORMATION**

This service will return all the studies that have data in the instance. Optional it will return information for a study with the study Prefix ID (Study ID) provided. The Study Prefix ID can be retrieved using this endpoint.

The following information is needed

**Endpoint URL:** [https://bricsnei-stage.cit.nih.gov/gateway/query-api/study](https://bricsnei-stage.cit.nih.gov/gateway/query-api/study)

**Parameters:**

| Headers | Response content type: application/json  
| --- | Content-Type: application/json  
| Authorization: Bearer + Token |

| Data | Optional: prefixedId = |
Figure 1: Example of Input

```python
url = "https://bricsnei-stage.cit.nih.gov/gateway/query-api/study"
headers = {
    'accept': 'application/json',
    'Content-type': 'application/json',
    'Authorization': 'Bearer ' + token
}

query = requests.get(url, headers=headers)
```

<table>
<thead>
<tr>
<th>Output Format</th>
<th>JSON</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Output Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Output Object</th>
<th>Required (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study Title</td>
<td>Yes</td>
</tr>
<tr>
<td>Study ID</td>
<td>Yes</td>
</tr>
<tr>
<td>Abstract</td>
<td>Yes</td>
</tr>
<tr>
<td>Principal Investigator</td>
<td>Yes</td>
</tr>
</tbody>
</table>
GET ALL STUDIES ASSOCIATED WITH A FORM STRUCTURE

Returns all the studies that have data submitted to the form structure

The following is needed

Endpoint URL: https://bricsnei-stage.cit.nih.gov/gateway/query-api/study/form?formName=

Parameters:

<table>
<thead>
<tr>
<th>Headers</th>
<th>Response content type: application/json</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Type: application/json</td>
</tr>
<tr>
<td></td>
<td>Authorization: Bearer + Token</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>Form Structure Short Name</th>
</tr>
</thead>
</table>


Figure 1: Example of Input

Figure 2: Example of Output
The Form Structure API uses the Study Prefix ID and returns the form structures that have data submitted against in a JSON format. To learn more about retrieving the Study Prefix ID, please refer to the section on the Study API endpoints.

GET FORM STRUCTURES FOR A STUDY

Returns all the form structures that have data submitted for the study.

The following is needed
**Endpoint URL:** https://bricsnei-stage.cit.nih.gov/gateway/query-api/form/study?prefixedId=

**Parameters:**

<table>
<thead>
<tr>
<th>Headers</th>
<th>Response content type: application/json</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Type: application/json</td>
</tr>
<tr>
<td></td>
<td>Authorization: Bearer + Token</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>STUDYID</th>
</tr>
</thead>
</table>


**Figure 1: Example of Input**

```python
url = "https://bricsnei-stage.cit.nih.gov/gateway/query-api/form/study?prefixedId="
headers = {
    'accept': 'application/json',
    'Content-Type': 'application/json',
    'Authorization': 'Bearer ' + token
}
studyid = input("Enter Study PrefixID")

query = requests.get(url + studyid, headers = headers)
```

**Output**

<table>
<thead>
<tr>
<th>Output Format</th>
<th>JSON</th>
</tr>
</thead>
</table>

**Output Description**

<table>
<thead>
<tr>
<th>Output Object</th>
<th>Required (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study Prefix ID</td>
<td>Yes</td>
</tr>
<tr>
<td>Form Structure Short Name</td>
<td>Yes</td>
</tr>
<tr>
<td>Form Structure Title</td>
<td>Yes</td>
</tr>
</tbody>
</table>
**DATA ELEMENTS API**

The data element API uses the form structure short name and returns the data elements within that form structure. The output will be in a JSON format and provide information about the data element such as the position in the form structure, the title and variable name.

The information about the data element is useful when filtering data in the Data API.

Below is information and examples for retrieving the data elements for that form structure.

**GET DATA ELEMENTS FOR A FORM STRUCTURE**

Return all data elements associated with the form structure.

The following is needed

**Endpoint URL:** https://bricsnei-stage.cit.nih.gov/gateway/query-api/dataElement/form/

**Parameters:**

<table>
<thead>
<tr>
<th>Headers</th>
<th>Content-Type: application/json</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Authorization: Bearer + <strong>Token</strong></td>
</tr>
</tbody>
</table>

| Data    | Form Structure Short Name |
```
GET Response

Form Structure Shortname", headers=headers)
```

**Figure 1: Example of Input**

```python
def get_form_strings(url):
    headers = {
        # 'accept': 'application/json',
        'Content-type': 'application/json',
        'Authorization': 'Bearer ' + token
    }

    print("Input Form Structure Short Name")
    formstructureshortname = input()

    dataelementapiquery = requests.get(url + formstructureshortname, headers=headers)
```

**Output**

```
Output Format
JSON
```

**Output Description**

<table>
<thead>
<tr>
<th>Output Object</th>
<th>Required (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Element Name</td>
<td>Yes</td>
</tr>
<tr>
<td>Form Structure Group Name</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Element Title</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Element Short Description</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Element Data Type</td>
<td>Yes</td>
</tr>
</tbody>
</table>
DATA API

The Data API allows users to retrieve data for one or more studies and form structures. Similar to the Query Tool, users are able to (1) Download data for form structures within a study, (2) Join form structures for one or more studies, (3) Filter on data elements with the advance Boolean Search.

Please note for the current release of King Kong, associated files and images are not downloadable.

Below are the endpoints and examples for retrieving data from studies and form structures.

GET DATA FROM MULTIPLE FORM STRUCTURES WITHOUT DOING JOINS

Returns data for multiple form structures without doing joins could also include study association if required.

The following is needed

Endpoint URL: https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/form/study

Parameters:

| Headers                  | Content-Type: application/json
|                         | Authorization: Bearer + Token
| Data                     | Form Structure Short Name
|                          | Optional: Study Prefix IDs
### GET Response

```python
response = requests.get("https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/form/study, headers=headers, json=data")
```

![Figure 1: Example of Input](image)

```python
multipleformsheader = {
    'Content-type': 'application/json',
    'Authorization': 'Bearer ' + token
}

multipleformurl = "https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/form/study"

multipleformsfilter = {
    "flattened": "false",
    "forexclude": [
        {
            "form": "AREDS2_AdverseEventReview",
            "studies": ["NEI BRICS-STUDY0000285"
        }
    ,
        {
            "form": "PROW1_Demo_2",
            "studies": ["EVIGENS-STUDY0000264"
        }
    ],
    "outputformat": "csv"
}

multipleformsquery = requests.post(multipleformurl, headers=multipleformsheader, json=multipleformsfilter)
```

```python
<Response [200]>
```

### Output

<table>
<thead>
<tr>
<th>Output Format</th>
<th>Zip Files</th>
</tr>
</thead>
</table>

### Output Description

<table>
<thead>
<tr>
<th>Output Object</th>
<th>Required (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV File of Data</td>
<td>Yes</td>
</tr>
</tbody>
</table>

![Figure 2: Example of Output](image)
GET DATA FROM MULTIPLE FORM STRUCTURES FOR THE GIVEN STUDIES

Returns data for multiple form structure to study associations, without doing joins

The following is needed

Endpoint URL: https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/study/form

Parameters:

<table>
<thead>
<tr>
<th>Headers</th>
<th>Data</th>
<th>GET Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content type: application/zip</td>
<td>Form Structure Short Name</td>
<td>response = requests.get(&quot;<a href="https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/form/study">https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/form/study</a>, headers=headers, json=data&quot;)</td>
</tr>
<tr>
<td>Content-Type: application/json</td>
<td>Study Prefix IDs</td>
<td></td>
</tr>
<tr>
<td>Authorization: Bearer + Token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Example of Input

```python
multipleformsstudy = {
    'accept': 'application/zip',
    'Content-Type': 'application/json',
    'Authorization': 'Bearer ' + token
}

multipleformsstudyurl = 'https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/study/Form'

multipleformsstudyfilter = {
    "limitseed": "false",
    "outputformat": "csv",
    "studyForms": [
        { "Form": [ "eyeGENE_Clinical" ], "study": "NEI_BRICS-STUDY0000207" },
        { "Form": [ "PROA_Bono_2" ], "study": "FVGGENF-STUDY0000304" }
    ]
}

multipleformsstudyquery = requests.post(multipleformsstudyurl, headers=multipleformsstudy, json=multipleformsstudyfilter)
multipleformsstudyquery
```
### Output

<table>
<thead>
<tr>
<th>Output Format</th>
<th>Zip Files</th>
</tr>
</thead>
</table>

### Output Description

<table>
<thead>
<tr>
<th>Output Object</th>
<th>Required (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV File of Data</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Figure 2: Example of Output

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
</table>

### GET DATA WITH FILTER AND JOINS

Returns data with filters and joins with up to five form structures. Data can be returned in two formats: text (csv) and JSON.

---

**The following is needed**

### Endpoint URL:

### Parameters:

| Headers | Response content type: application/csv or application/json  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Type: application/json</td>
</tr>
<tr>
<td></td>
<td>Authorization: Bearer + Token</td>
</tr>
</tbody>
</table>

| Data | Filter on data elements, form structure and studyID |
| GET Response | response = requests.get("https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/bulk/form/study, headers=headers,json=data") |

---

### Figure 1: Example of Input CSV
Figure 2: Example of JSON Input

```
queryurl = "https://bricsnei-stage.cit.nih.gov/gateway/query-api/data/json"
headers = {
    'accept': 'application/json',
    'Content-type': 'application/json',
    'Authorization': 'Bearer ' + token
}

query = requests.post(queryurl, headers=headers, json=genomicsfilter2)

<Response [200]>
```

**Output**

<table>
<thead>
<tr>
<th>Output Format</th>
<th>Text</th>
</tr>
</thead>
</table>

16 • BRICS USER MANUAL © 2019 National Institutes of Health/CIT – Proprietary and Confidential
Output Description

<table>
<thead>
<tr>
<th>Output Object</th>
<th>Required (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV Format with Data</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 3: Example of Output

dataset = query.text
text2df = StringToDataFrame(dataset)
real_data = pd.read_csv(text2df, sep="\", )
real_data.head()

C:\Users\Name\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\formats\csv.py:4398: DeprecationWarning: Columns (24) have mixed types. Specify dtype option on import or set low_memory=False.

Interactivity:interactivity, compiler:compiler, text=/news2.txt

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NLI_UVivingIPIO</td>
<td>NLI_BRICS-DATE000G591</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NLI_UVivingIPIO</td>
<td>NLI_BRICS-DATE000G591</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NLI_UVivingIPIO</td>
<td>NLI_BRICS-DATE000G591</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NLI_UVivingIPIO</td>
<td>NLI_BRICS-DATE000G591</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NLI_UVivingIPIO</td>
<td>NLI_BRICS-DATE000G591</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td>NLI_UVivingIPIO</td>
<td>4327</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 rows x 60 columns

(len(real_data)

Output Format

| Output Format | JSON |

Output Description

<table>
<thead>
<tr>
<th>Output Object</th>
<th>Required (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUID</td>
<td>Yes</td>
</tr>
<tr>
<td>Form Structure Shortname</td>
<td>Yes</td>
</tr>
<tr>
<td>Study ID</td>
<td>Yes</td>
</tr>
<tr>
<td>Dataset ID</td>
<td>Yes</td>
</tr>
<tr>
<td>Form Structure Repeatable Group</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Elements and Associated Data</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Figure 4: Example of JSON Output of data

```json
jsondata = query_json()

jsondata

[{
    "guid": "NIL_EWEV49P8",
    "form": {
        "name": "eyeGENE Genomics V1.2",
        "studyId": "EYEGENE-01799",
        "reportDate": "2020-03-01 12:00:00",
        "eyeGENEDoID": "T42D",
        "eyeGENEFamilyID": "0134"},
    "name": "Genetic Information",
    "date": {
        "genetestreportdate": "2029-11-19T00:00:00",
        "genetestresultdate": "2029-11-19T00:00:00",
        "genetestresultreportdate": "2029-11-19T00:00:00"},
    "CLINICALNAME": "PreventionGenetics",
    "CLINICALNAMEID": "5200005312",
    "EyeConditionFamilyName": "Direct Sequencing",
    "EyeConditionFamilySymbol": "ABCA1",
    "EyeConditionFamilyGene": "ABCA1",
    "EyeConditionFamilyGeneSymbol": "ABCA1",
    "Instrument": "1.5A",
    "GeneVariantIndicator": "Yes",
    "HVSsNoSampleDBA": "c.e009t0",
    "HVSsNoSampleProven": "p.e123313le",
    "GeneVariantType": "unknown",
    "GeneVariantInterType": "unknown",
    "ReferenceVariantID": "донг:1782124",
    "EyeGENEtestreportfilename": "Yes",
    "EyeGENEtestreportfilestatus": """
}]
```
APPENDIX

Attached are Sample API Scripts:

<table>
<thead>
<tr>
<th>Script Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRICSAPI_UserGuide</td>
<td>Provides examples of the API Endpoints</td>
</tr>
<tr>
<td>API_Case1</td>
<td>From the join of the two form structures, eyeGeneDemographics and eyeGeneGenomics, provide the GUIDs with one gene and one or more gene variant types.</td>
</tr>
<tr>
<td>API_Case2</td>
<td>The purpose of the script is to provide the list of GUIDs from the eyeGeneGenomics form structure that have one gene and one or more gene variant types.</td>
</tr>
<tr>
<td>API_Case3A</td>
<td>The purpose of the script provides the GUIDs for a gene, but does not include a gene variant type.</td>
</tr>
<tr>
<td>API_Case3B</td>
<td>The purpose of the script provides the GUIDs with multiple genes and gene variant type.</td>
</tr>
<tr>
<td>API_GenesandExcludeGeneVariant</td>
<td>The purpose of the script provides the list of GUIDs with two genes and a gene variant type, but excludes additional gene variant type.</td>
</tr>
<tr>
<td>API_Genevarianttype</td>
<td>The purpose of the script is to provide the list of GUIDs with two genes and a gene variant interpretation type.</td>
</tr>
<tr>
<td>API_GUIDsnotinGenomics</td>
<td>The purpose of this script is to return the GUIDs that exist in the eyeGeneDemographics data, but not in the eyeGeneGenomics data.</td>
</tr>
<tr>
<td>HGNCGeneSymblandGeneIndicator</td>
<td>The purpose of this script is to return all GUIDs that have two genes and the gene indicator is “yes”.</td>
</tr>
</tbody>
</table>